AUSTRALIA

Patent Office
Canberra

CERTIFIED COPY OF PRIORITY DOCUMENT

I, LEANNE MYNOTT, TEAM LEADER EXAMINATION SUPPORT AND SALES hereby certify that annexed is a true copy of the Provisional specification in connection with Application No. PQ3217 for a patent by CANON KABUSHIKI KAISHA filed on 30 September 1999.

I further certify that pursuant to the provisions of Section 38(1) of the Patents Act 1990 a complete specification was filed on 25 September 2000 and it is an associated application to Provisional Application No. PQ3217 and has been allocated No. 61287/00.

WITNESS my hand this
Fourth day of October 2000

[illegible]

LEANNE MYNOTT
TEAM LEADER EXAMINATION
SUPPORT AND SALES

Appln. No.: 09/668,465
Filed: September 25, 2000
Inv.: Ding Mei Zhang, et al.
Title: Television Program Recommendation System

S & F Ref: 472595

ORIGINAL

AUSTRALIA

Patents Act 1990

PROVISIONAL SPECIFICATION FOR THE INVENTION ENTITLED:

A Television Program Recommendation System

Name and Address of Applicant:	Canon Kabushiki Kaisha, incorporated in Japan, of 30-2, Shimomaruko 3-chome, Ohta-ku, Tokyo, 146, JAPAN
Inventor(s) Name(s):	Dong Mei Zhang, Wai Yat Wong, Mikhail Propopenko, Farhad Fuad Islam, Ryszard Kowalczyk, Michael Alexander Oldfield, Marc Butler and Paul Trayers

This invention is best described in the following statement:

A TELEVISION PROGRAM RECOMMENDATION SYSTEM

Field of the Invention

The present invention relates to television program selection and more particularly to a system for providing recommendations for television program selection.

Description of Background Art

Watching television is an everyday activity in a large number of households, and provides for a source of entertainment for a range of program content such as sport and movies, as well as news and actuality programs (eg. documentaries and lifestyle).

Traditionally, a viewer would consult a printed television schedule listing in a local newspaper or periodical to find a desirable program to watch. More recently, entertainment program guides have been made available in electronic form via the Internet or World Wide Web. However, with the introduction of satellite receivers for television and cable television, the number of television channels available to the viewer has increased dramatically. This has made the task of selecting a television program to watch using either printed program listings or on-screen Electronic Programming Guides (EPG) very involved and time consuming.

Program listings that are ordered by characteristics including category, time or performers, can assist the viewer in making a selection, but such remains a time consuming task. Typically, there may only be a few programs of interest to the viewer out of the vast number of available programs.

Systems are available that monitor those categories of programs a viewer most frequently watches and to provide recommendations based on the most frequently watched category. Such arrangements suffer from the disadvantage of not recognising that selection by the viewer may depend not only on a single program characteristic, but on a combination of characteristics.

Disclosure of the Invention

It is an object of the present invention to substantially overcome, or at least ameliorate, one or more disadvantages of existing arrangements.

According to a first aspect of the invention, there is provided a method of forming a database of user viewing characteristics to be used in enabling a subsequent selection, by a corresponding user, of a program for viewing in a television system in which title information and characteristics of programs to be transmitted in future is transmitted in advance as at least one program guide list, the method comprising the steps of:

recording a plurality of characteristics associated with each program viewed by said user;

forming sets of said characteristics, each said set comprising at least two of said characteristics; and

associating at least each set with an ordered value representative of user's desire to view a particular program;

wherein upon entry of a user request for a program recommendation, performing a search of each of said program guide lists for programs with characteristics that best match said sets and notifying said user of an availability of programs that best match said sets as program recommendations.

According to another aspect of the invention, there is provided a recommendation system for enabling a selection by a user of a program for viewing in a television system in which title information and characteristics of programs to be transmitted in future is transmitted in advance as at least one program guide list, the method comprising the steps of:

memory means for recording a plurality of characteristics associated with each program viewed by said user;

processing means for forming sets of said characteristics, each said set comprising at least two of said characteristics and for associating each said set with an ordered value representative of said user's desire to view a particular program;

searching means for performing a search of each of said program guide lists for programs with characteristics that best match said sets; and

on-screen display means for notifying said user of an availability of programs that best match said sets as program recommendations upon entry of a user request for program recommendations.

According to another aspect of the invention there is provided a computer program product including a computer readable medium having recorded thereon a computer program for implementing the method described above.

Brief Description of the Drawings

A number of preferred embodiments of the present invention will now be described with reference to the drawings, in which:

Fig. 1A is a schematic representation of a system according to an embodiment of the invention;

Fig. 1B is a detailed representation of an avatar agent of the system in Fig. 1A;

Fig. 2 is an extract from a typical Electronic Program Guide;

Fig. 3 is an illustration of an animated character displayed on a display screen of the system in Fig. 1A;

Fig. 4 is an example of how recommendations may be presented to a viewer by the system in Fig. 1A;

Fig. 5 is an example of a selection made from the recommendations in Fig. 4;

Fig. 6 is an example of a viewer profile database;

Figs 7A to 7K are flow diagrams of a method performed by a learning module of the avatar agent illustrated in Fig. 1A, to learn common features from selections made by the viewer;

Fig. 8 is an illustration of the results obtained by the method in Figs 7A to 7K;

Fig. 9 is a flow diagrams of a method performed by a recommendation module of the avatar agent illustrated in Fig. 1B, to use the results from the learning module to make recommendations of programs to watch to the viewer; and

Fig. 10 is a schematic block diagram of a general purpose computer upon which an embodiment of the present invention can be practiced.

Detailed Description including Best Mode

Fig. 1A shows a system 50 for automatically suggesting suitable programs to a viewer from a large number of available programs. The system 50 comprises a digital television (DTV) 10 connected through interconnection 25 to a "set top" box 20. The "set top" box 20 is also connected through a gateway 41 to an external network 42. A number of content servers 43 and Electronic Program Guide (EPG) databases 22 are connected to the external network 42. The content servers 43 are typically provided by content providers and contain multimedia content including movies and television programs.

The available programs are listed in the EPG databases 22. Each of the content providers may maintain an associated EPG database 22. The available programs listed in the EPG databases 22 may be linked to the corresponding multimedia content in the content servers 43 by means of a program identifier. An extract from a typical EPG database 22 is shown in Fig. 2. The EPG database 22 has a number of program entries 60, each of which may include a number of attributes 61 with values 62. The attributes 61 may include:

- EPG ID which is a unique number per entry;
- Program ID which is unique for each program resulting in re-runs having the same Program ID;
- Category ID, for example '01' for art, '16' for drama, '50' for Movies and Series, '75' for sport etc.;
- Subcategory ID, for example, in the movies category '50', the subcategory '001' for action/adventure, '064' for comedy, '074' for crime etc;
- Title;
- EPG Channel;
- Day;
- Date;
- Start time;
- Duration; and
- Year of Make.

A DTV-agent system 21 is preferably formed within the "set top" box 20. In use, the viewer interacts with the DTV-agent system 21 using a remote control device 30. The DTV-agent system 21 may alternatively be integrated into the DTV 10 or incorporated into a personal computer 100 such as that seen in Fig. 10 and appropriately interfaced with the DTV 10.

In the preferred embodiment of the invention, the DTV-agent system 21 includes a DTV-agent 36 for controlling the DTV 10, and a number of avatar-agents 37, each avatar-agent 37 representing a particular viewer. An Inter-agent-server (IAS) 35 manages all inter-agent communications and also interfaces with the gateway 41. The DTV-agent system 21 further includes a number of viewer profile databases 23 accessed by the avatar-agents 37. The functions of the DTV-agent 36 include:

- the provision of a graphical user interface via a display screen 11 of the DTV 10 by which recommendations are made to the viewer and to enable the viewer to make selections;
- to control the functionality of the DTV 10 including interacting with a television tuner module (not illustrated) thereof to select a channel for viewing from the plurality of available television channels;
- to gather viewer selections and delivers them to the avatar agents 37;

Each of the avatar-agents 37 is associated with a single viewer and a corresponding one of the viewer profile databases 23. As seen in Fig. 1B, each avatar-agent 37 includes an avatar manager 38 that maintains control of the particular avatar-agent 37. The avatar manager 38 accesses the EPG databases 22 through the IAS 35 to obtain television program information and the associated viewer profile database 23 to build and maintain the viewer profile. The avatar manager 38 is also responsible for sending messages to, and receiving messages from, the DTV-agent 36 through the IAS 35. Within the avatar-agent 37, the avatar manager 38 is also responsible for interfacing with each of a learning module 39 and a recommendation module 40. One task of the avatar-agent 37 is to build and maintain the viewer profile database 23 based on selections made by the viewer and processed by the learning module 39. The avatar-agent 37 then also uses the viewer profile database 23 to provide recommendations to the viewer by processing performed in the recommendation module 40. The recommendations are for the most suitable viewing options available and are provided to the viewer via the display screen 11.

Referring to Tables 1 and 2, examples of typical steps for use of the system 50 will now described.

TABLE 1

Step	Message	From	To	Internal processing
1#				Viewer switches DTV 10 on.
2				Animation characters 12 are displayed on the display screen 11. Viewer chooses animation character 12.
3	SetAnimationSelection	DTVA	AA	
4#	SendUserData	AA	DTVA	
5				Viewer interactively provides personal information and a viewing history based on prompts displayed on the display screen 11.
6 *	SetNewUserData	DTVA	AA	
7#				Viewer asks for recommendations 67.
8				Prompt viewer for time and date.
9 *	SendRecommendationRequest	DTVA	AA	
10#	SetRecommendations	AA	DTVA	
11				EPG and preview data retrieved.
12				Preview data are displayed on display screen 11. Viewer makes a selection 68.
13				If the viewer asks for more recommendations 67, go to sequence step 9.
14 *	SetProgramSelection	DTVA	AA	

Note: When the sequence shown in Table 1 reaches a *, the DTV-agent 36 enters an "idle" mode, waiting for messages from avatar agent 37. In this mode, the DTV 10 displays a television program (on a channel most recently selected by the viewer). Sequence points indicated by a hash symbol (#) can be accessed at any time (i.e. asynchronously).

Table 1 illustrates a sequence of events and resulting messages to and from the DTV-agent 36 in which:

1. The user switches the DTV 10 'ON' using the remote control 30.
2. The viewer needs to identify himself/herself to the DTV-agent system 21. Referring to Fig. 3, this may be done by selecting an animation character 12

from a set of characters generated by the DTV-agent 36 and displayed on the display screen 11. The selection may be performed using the remote control 30. In the preferred embodiment, there may be three types of characters:

(i) Un-touched characters, meaning those that are not associated with any viewer;

(ii) Dormant characters, meaning those that are associated with a viewer but that particular viewer is currently not engaged; and

(iii) An active character 12, identifying that the associated viewer is presently selected. No active character will be present when the DTV 10 has just been turned 'ON'.

3. A SetAnimationSelection message is sent to the avatar agents 37, activating only the avatar-agent 37 associated with the identified viewer.

4. A SendUserData message is received from the avatar agent 37, instructing the DTV agent 36 to collect viewer data.

5. The DTV-agent 36 controls the display screen 11 of the DTV 10 for the collection of viewer data. The viewer provides the data using the remote control 30. The first time that a character 12 is selected, the system 50 assumes that it is a new viewer using the system 50 and therefore an unregistered viewer. The avatar agent 37 has no viewer profile database for the currently unregistered viewer.

Additionally, during the registration process, the viewer may be requested by the DTV-agent 36 for details of programs the viewer can remember viewing in the past such as the name of dramas previously enjoyed, the name of a preferred news caster, preferred music program, etc. This may be done by displaying previous season's program listings on the display screen 11 and enabling the viewer to select preferred programs using the remote control 30.

All viewers may then be requested by the DTV-agent 36 to select one of a number of moods currently being experienced by the viewer, again by making a selection with the remote control 30. Examples of such moods include happy, sad, neutral etc. A password may also be requested from the viewer when the program content is password-protected (eg. adult content).

6. The viewer data are formatted and sent in a SetNewUserData message to the avatar-agent 37. The DTV agent 36 enters into the idle mode.
7. The viewer uses the remote control 30 to request from the DTV-agent system 21 a list of recommended television programs.
8. The DTV-agent 36 prompts the viewer for a time and date for which recommendations should be provided. Alternatively, recommendations may automatically be for television programs due to start in the following predetermined number of minutes.
9. The time and date are formatted and sent to the avatar-agent 37 in a SendRecommendationRequest message. The DTV agent 36 enters into the idle mode.
10. The avatar agent 37, and in particular the recommendation module 40, determines viewing recommendations for viewing from the EPG databases 22 that best match the viewer's preferences for the given timeslot and the viewer mood. The avatar agent 37 sends these viewing recommendations in a SetRecommendations message to the DTV agent 36.
11. Based on the recommended programs received from the avatar agent 37, data are retrieved from the EPG databases 22 and one or more preview databases

(not shown). The preview databases may form part of the EPG databases 22 or alternatively, the content servers 43 and provide a brief example or preview of the content of a particular program.

12. The DTV-agent 36 formats the viewing recommendations and displays the recommendations on the display screen 11. As illustrated in Fig. 4, the viewing recommendations 67 are preferably provided to the viewer on the display screen 11 by displaying a set of previews. Alternatively, a bitmap image or a title only is displayed on the display screen 11, each of which representing a viewing recommendation 67. Additional information may also be displayed associated with each of the recommendations 67. The viewing recommendations 67 may also be displayed in a manner to reflect the program attributes. For example, all the viewing recommendations 67 having a common category may be displayed together. The viewer views the viewing recommendations 67 and selects one of the viewing recommendations 67 using the remote control 30. The DTV-agent 36 interacts with the DTV 10, which responds by tuning into a receiver channel on which the selected program is broadcast. The selected program 68 is then displayed on the full display screen 11 as shown in Fig. 5. Alternatively, the viewer may ask for other set of recommendations 67.
13. If the viewer has asked for more recommendations 67, the DTV agent 36 reverts to sequence step 9.
14. Information about the selected program is sent to the avatar agent 37 in a SetProgramSelection message. The DTV-agent 36 enters its idle mode while the selected program 68 is displayed on the display screen 11.

TABLE 2

Step	Message	From	To	Internal processing
1#	SetAnimationSelection	DTVA	AA	
2				Retrieve information from viewer profile database 23 to establish whether the viewer has profile data. If not, go to step 3.
*				
3				Create viewer profile database 23.
4	SendViewerData	AA	DTVA	
5	SetNewViewerData	DTVA	AA	
6				Retrieve TV program entries 60 corresponding to historical viewing from EPG database 23.
7				Send viewer data and historical program data to learning module 39.
8				Learning module 39 constructs GPList.
9				Receive GPList from learning module 39.
10				Update viewer profile database 23.
*				
11#	SendRecommendationRequest	DTVA	AA	
12				Retrieve viewer profile and EPG information from databases 23 and 22. Retrieve television program information corresponding to requested time/date from EPG database 22.
13				Send viewer data and television program information to recommendation module 40.
14				Recommendation module 40 constructs a list of recommended television programs 67.
15				Receive program recommendations 67 from recommendation module 40.
16	SetRecommendations	AA	DTVA	
*				
17#	SetProgramSelection	DTVA	AA	
18				Retrieve viewer profile and program data from databases 23 and 22.
19				Send viewer data and new program data to learning module 39.
20				The learning module 39 updates the GPList.
21				Receive GPList from learning module 39.
22				Update viewer profile database 23.
*				

Note: When the sequence shown in Table 1 reaches a *, the avatar-agent 37 enters an "idle" mode, waiting for messages from the DTV-agent 36. In this mode, background processing by the learning module 39 and the recommendation module 40 may occur. Sequence points indicated by a hash symbol (#) can be accessed at any time (i.e. asynchronously).

Table 2 illustrates a sequence of events and resulting messages to and from the avatar-agents 37 in which:

1. A SetAnimationSelection message is received from the DTV-agent 36 by the avatar-agent 37 associated with the selected animation character 12. The other avatar-agents 37 remain in the idle mode.
2. The avatar agent 37 determines whether the selected character 12 corresponds to a viewer known to the DTV-agent system 21, or to a new viewer. If the viewer is not known to the DTV-agent system 21, the next sequence step is step 3. Otherwise the avatar manager 38 enters its idle mode.
3. A profile for the new viewer is created in the viewer profile database 23.
4. A SendViewerData message is sent to the DTV agent 36.
5. A SetNewViewerData message is received from the DTV agent 36. This contains static viewer attributes and a list of programs that the viewer has viewed in the past.
6. The avatar-agent 37 retrieves the program entries from the EPG database 22 of the programs that the viewer has identified in step 5.
7. The viewer's data is formatted and sent to the learning module 39.
8. The learning module 39 constructs a Generalised Pattern List (GPList).
9. The GPList is passed from the learning module 39 to the avatar manager 38.
10. The avatar manager 38 stores the GPList in the viewer profile database 23. The avatar manager 38 enters its idle mode.

11. A SendRecommendationRequest message is received from the DTV agent 36. The viewer has requested that viewer recommendations 67 for a particular day and time be provided.
12. The avatar-manager 38 retrieve viewer profile data from viewer profile database 23 and retrieve television program information corresponding to requested time/date from the EPG database 22. The data is formatted for use by recommendation module 40.
13. The viewer profile data and the program information are sent to the recommendation module 40 by the avatar manager 38.
14. The recommendation module 40 constructs a list of programs to be recommended to the viewer.
15. The list of recommended programs is sent from the recommendation module 40 to the avatar manager 38.
16. A SetRecommendations message, containing a list of recommended programs is sent to the DTV agent 36. The avatar manager 38 enters its idle mode.
17. A SetProgramSelection is received from the DTV agent 36, indicating that the viewer has selected a particular program 68.
18. The program entry 60 about the selected program 68 is retrieved from the EPG database 22. The viewer profile is also retrieved from the viewer profile database 23.
19. Viewer history and the selected program information is sent to the learning module 39 for updating of a Case-file.
20. The learning module 39 updates the GPList to take account of the selected program 68.
21. The updated GPList is received by the avatar manager 38 from the learning module 39.

22. The updated GPList is used to update the viewer profile database 23. The avatar manager 38 enters its idle mode.

The operations within the learning module 39 will now be described in more detail. Each program is associated with a number of features (fi), each feature representing a unique attribute and attribute value pair. The attributes may include the program attributes 61 from the EPG database 22 (illustrated in Fig. 2) as well as viewer attributes when the selection was made. The viewer attributes may include for example the mood of the viewer. Each attribute has a number of possible values. An example of an attribute-value pair is *Category* = 'drama' where *Category* is the attribute and 'drama' is the attribute value.

Each selection 68 made by the viewer is termed a case (Cj), and is defined by a set of features (fi). For example, a case may be (*Category* = 'drama', *Sub-category* = 'comedy-drama', ..., *EPG channel* = 'Nhk', ...). The cases (Cj) may have different number of features (fi), dependent on the information available. All cases are stored in the viewer profile database 23 in a case file for use by the learning module 39.

The learning module 39 operates to identify shared or generalisation patterns of features (fi) from a number of cases (Cj) from the case file. The learning module 39 takes the case file as input and generates a Generalisation Pattern List (GPList), stored also in the viewer profile database 23. The GPlist contains all generalisation patterns from the case file. Each generalisation pattern in the GPList may be represented as follows:

([intersection], occurrence),

wherein intersection indicates a set of the features (fi) that are shared by different cases (Cj). Occurrence indicates the number of cases that share such an intersection. For example, from the following cases:

C1 = (f1, f2, f3, f4, f7);

C2 = (f1, f2, f5, f6, f7); and

C3 = (f3, f5, f6, f8)

the GPList generated from the three cases has three items:

([f1, f2, f7], 2);

([f3], 2); and

([f5, f6], 2)

The first GPList entry above arises from the features f1, f2 and f7 occurring both in cases C1 and C2. The other entries are derived in a similar manner.

Each time new cases (Cj) become available through selections by the viewer of programs to watch, those new cases are added to the case file and a new GPList is generated by the learning module 39. Assume for example that the case file contains the following three cases:

C1 = (*Category* = 'drama', *Sub-category* = 'comedy-drama',*EPG channel* = 'Nhk');

C2 = (*Category* = 'drama', *Sub-category* = 'war', *EPG channel* = 'Nhk', *StartTime* = '21:00'); and

C3 = (*Category* = 'sports', *EPG channel* = 'Wow', *StartTime* = '8:15', *Day* = 'Sat').

The GPList generated by the learning module would only include:

([*Category* = 'drama', *EPG channel* = 'Nhk'], 2)

When new viewer selections, and therefore further cases, became available, for example

C4 = (*Category* = 'drama', *Sub-category* = 'Talk', *EPG channel* = 'Nhk'); and

C5 = (*Category* = 'sports', *Sub-category* = 'Current affairs', *StartTime* = '10:00', *Day* = 'sat')

cases C4 and C5 are added to the case file and the GPList is recalculated by the learning module 39 to contain the following entries:

([*Category* = 'drama', *EPG channel* = 'Nhk'], 3)

([*Category* = 'sports', *Day* = 'Sat'], 2)

A procedure **MAIN** for performing the core function in the learning module 39 is shown in Fig. 7A. Three data structures are used in the procedure namely, the Case-file which contains a number of cases (Cj), the GPList which keeps all generalisation patterns, and an Examined-case-list which contains cases that are already processed. Initially, upon initiation of the procedure **MAIN** when a viewer requested for recommendations, the GPList and the Examined-case-list are empty.

The procedure **MAIN** starts in step 200 by first obtaining a list of new cases in step 201 and the Case-file from the viewer profile database in step 202. A sub-routine **UPDATE-CASE-FILE** is called in step 203 for updating the Case-file by adding the new cases. The sub-routine **UPDATE-CASE-FILE** starts at step 212 in Fig. 7B.

Intersections for each case of the Case-file with the current GPList is obtained in the remaining steps 204 to 211. A first entry from the Case-file is taken in step 204 and used as an input case when calling sub-routine **GEN-CASE-GPList** in step 205. Sub-routine **GEN-CASE-GPList** finds the generalization patterns between the input-case and items in the GPList and starts at step 220 in Fig. 7C. Sub-routing **GEN-CASE-EXAMINED-CASES** is called in step 206 for finding generalization patterns between the input-case and the cases in the Examined-case-list. Sub-routing **GEN-CASE-EXAMINED-CASES** starts at step 310 in Fig. 7H. In step 207, the input-case is added to the Examined-case-list. Cases (Cj) are therefore progressively moved from the Case-file to the Examined-case-list until the Case-file is empty. Step 208 determines whether Case-file has any more entries. If there are any remaining entries in the Case-file, the procedure **MAIN** proceeds to step 209 where the next entry from Case-file is used as input-case and the procedure **MAIN** continues to step 205.

After step 208 determines that the Case-file is empty, the procedure **MAIN** produce generalization patterns in the GPList as an output in step 210 and ends in step 211.

Referring now to Fig. 7B wherein the sub-routine **UPDATE-CASE-FILE** is shown. This sub-routine includes the new cases produced as input in step 201 in procedure **MAIN** and into them into the current Case-file. It receives the entries from the list of new cases in step 213 and taking a next new case from the list of new cases in step 214, the next new case is added onto the end of the Case-file. Step 216 determines whether all the new cases has been dealt with and proceeds to steps 217 and 215 where the next entry is added to the Case-file. An updated Case-file is produced as output in step 219 and the sub-routine **UPDATE-CASE-FILE** ends in step 219.

Referring to Fig. 7C , the sub-routine **GEN-CASE-GPLIST** is to find all generalization patterns between the input-case and all items in the GPList. It further updates the GPList with all new generalization patterns between the input-case and the current GPList.

GEN-CASE-GPLIST starts in step 220 and receives the input case and the current GPList as inputs in step 221. It is determined in step 222 whether the current GPList is still empty. If the current GPList is empty, then no generalization patterns between the input-case and the GPList can exist and the sub-routine returns in step 236.

With items in the GPList, the sub-routine continues to step 223 wherein all generalization patterns between the input-case and all items in the GPList are found by calling sub-routine **G_List_GEN**. The generalization patterns are put in a G_List, which contains potential generalization patterns between the input-case and the GPList. The subroutine **G_List_GEN** starts at step 240 in Fig. 7D.

It is determined in step 224 whether the G_List is empty. If the G_List is empty, then no generalization patterns between the input-case and the GPList was found by sub-routine **G_List_GEN** and the sub-routine **GEN-CASE-GPLIST** returns in step 236.

With items in the G_List, the sub-routine **GEN-CASE-GPLIST** continues to step 225 where subroutine **UG_List_GEN** is called. Subroutine **UG_List_GEN** starts at step 260 in Fig. 7E, and forms a unique generalisation pattern list, UG_List, from the G_List.

A first item from the UG_List, UG_Item, is retrieved and the intersection, First_Intersection, from the UG_Item is extracted in step 226. A first item from the GPList, GPList_Item, is retrieved and the intersection, Second_Intersection, from the GPList_Item is extracted in step 227.

Step 228 calls sub-routine **INT_MATCH** to determine whether First_Intersection and Second_Intersection match. Sub-routine **INT_MATCH** starts at step 280 in Fig. 7F. If step 228 finds that First_Intersection and Second_Intersection match, the occurrence of GPList_Item is made equal to that of UG_Item in step 229. The sub-routine continues to step 233.

If step 228 finds that First_Intersection and Second_Intersection do not match, then step 230 determines whether all items of the GPList have been considered. If items remain, the next item in the GPList is retrieved with it's intersection as Second_Intersection in step 231, before step 228 again determines whether First_Intersection and Second_Intersection match. If step 230 determined that all items in the GPList have been considered, then the UG_Item is added to the GPList in step 232 and the sub-routine continues to step 233.

If step 233 determines that all items in the UG_List have not been considered, then the next item in the UG_List is retrieved with it's intersection as First_Intersection in step 234 and followed by step 227. Alternatively, if step 233 determines that all items in the UG_List have been considered, the sub-routine **GEN-CASE-GPLIST** outputs the GPList in step 235 and returns in step 236.

Referring to Fig. 7D, sub-routine **G_List_GEN** is described, which gets all generalisation patterns between the input-case with all items in the GPList. Starting at step 240 , it uses as inputs the input-case and the GPList. The inputs are obtained in step 241. A first generalisation pattern, GPList_Item is obtained from the GPList in step 242, a first feature from the input case is retrieved in step 243 and a first feature from the intersection part of the GPList_Item is retrieved in step 244. Step 245 determines whether the retrieved feature from the input-case is the same as the retrieved feature from the GPList_Item. If no match is found in step 245, step 246 determines whether all features from the GPList_Item has been dealt with. With more features in the GPList_Item remaining, step 255 retrieves a next feature from the GPList_Item and continues to step 245. If all the features in the GPList_Item were considered, the sub-routine continues to step 247.

If step 245 responds in the affirmative, step 252 determines whether a new generalization pattern has been created and create one in step 253 if required, or if it was already done, proceeds to step 254, where the shared feature is added to the new generalization pattern. The sub-routine continues to step 247 where it is determined whether all features from the input-case have been dealt with. If there are remaining features in the input-case, step 256 retrieves the next feature from the input-case.

If all the features from the input-case were considered, step 248 determines whether a new generalisation pattern was created in step 252. If it is not so, then the sub-routine continues to step 251. However, if a new generalisation pattern was created, the sub-routine continues to step 249 where **[I am not clear on what happen here. CISRA to advise]**. In step 250 the new generalisation pattern is added to the G_List.

Step 251 determines whether all items from the GPList have been considered. If an item remains in the GPList, the sub-routine continues to step 257 where the next item in the GPList is retrieved and step 243 is executed. Alternatively, with all items in the GPList considered, the sub-routine G_List_GEN returns in step 259 after producing the new G_List as output in step 258.

Referring to Fig. 7E, a sub-routine **UG_List_GEN** is shown that procedure forms a unique generalisation pattern list, UG_List. Starting in step 260, the sub-routine receives in step 261 the G_List as input. In step 262 the first generalisation pattern is copied from the G_List into the UG_List. A first item from the G_List, G_List_Item, is retrieved and the intersection, First_Intersection, from the G_List_Item is retrieved in step 263. A first item from the UGList, UGList_Item, is retrieved and the intersection, Second_Intersection, from the UGList_Item is retrieved in step 264.

Step 265 calls sub-routine **INT_MATCH** to determine whether First_Intersection and Second_Intersection match. Sub-routine **INT_MATCH** starts at step 280 in Fig. 7F. If First_Intersection and Second_Intersection match, the higher occurrence of the two items, G_List_Item and UG_List_Item, is determined and saved as the occurrence of UG_List_Item in steps 269 and 270. The sub-routine continues to step 268.

If First_Intersection and Second_Intersection do not match, step 266 determines whether all items of the UG_List have been considered. If items remain, the next item in the UG_List is retrieved with its intersection as Second_Intersection in step 271, before step 265 again determines whether First_Intersection and Second_Intersection match. If step 266 determined that all items in the UG_List have been considered, then the G_List_Item is added to the UG_List in step 267 and the sub-routine continues to step 268.

If step 268 determines that all items in the G_List have not been considered, then the next item in the G_List is retrieved with it's intersection as First_Intersection in step 263 and followed by step 264. Alternatively, if step 268 determines that all items in the

G_List have been considered, the sub-routine **UG_LIST_GEN** outputs the UG_List in step 273 and returns in step 274.

Referring to Fig. 7F, subroutine **INT_MATCH** is shown which check whether the First_Intersection and the Second_Intersection, obtained as inputs in step 281, are the same. Step 282 retrieves the first feature from First_Intersection, named First-feature, followed by step 283 where the first feature from Second_Intersection, named Second-feature, is retrieved. Step 284 calls sub-routine **FEATURE-SAME** to determine whether First-feature is the same as Second-feature. Sub-routine **FEATURE-SAME** starts at step300 in Fig. 7G. If they are the same, then step 285 determines whether all the features of First_Intersection have been considered. If this is affirmative, then the sub-routine continues to step 286. If not, then step 296 retrieves the next feature from First_Intersection, names it First-feature and continues to step 283. However, if step 284 determined that First-feature is not the same as Second-feature, then step 290 determines whether all the features of the Second_Intersection have been considered. If this is affirmative, then the sub-routine **INT_MATCH** returns a "NO" in step 292. If features of the Second_Intersection remain, then step 291 retrieves the next feature from Second_Intersection, names it Second-feature and continues to step 284.

Step 286 retrieves the first feature from Second_Intersection, named Second-feature, followed by step 287 where the first feature from First_Intersection, named First-feature, is retrieved. Step 288 calls sub-routine **FEATURE-SAME** to determine whether First-feature is the same as Second-feature. If they are the same, then step 289 determines whether all the features of Second_Intersection have been considered. If this is affirmative, then the sub-routine **INT_MATCH** returns a "YES" in step 298. If not, then step 297 retrieves the next feature from Second_Intersection, names it Second-feature and continues to step 287. However, if step 288 determined that First-feature is not the same as Second-feature, then step 293 determines whether all the features of the First_Intersection have been considered. If this is affirmative, then the sub-routine **INT_MATCH** returns a "NO" in step 295. If features of the First_Intersection remain, then step 294 retrieves the next feature from First_Intersection, names it First-feature and continues to step 288.

Referring now to Fig. 7G, sub-routine **FEATURE-SAME** is described, which checks whether two given features are the same. Therefore, in step 301 it receives as inputs, First-feature and Second-feature. Step 302 gets the attribute of First-feature, named First-f-attribute, and step 303 gets the attribute of Second-feature, named Second-f-attribute. Step 304 gets the value of First-feature, named First-f-value, and step 305 gets the value of Second-feature, named Second-f-value.

Step 306 determines whether First-f-attribute is the same as Second-f-attribute, and step 307 determines whether First-f-value is the same as Second-f- value. If the answers of both steps 306 and 307 are affirmative, then sub-routine **FEATURE-SAME** returns a "YES" in step 309. If any one of the answers of steps 306 and 307 is negative, then sub-routine **FEATURE-SAME** returns a "NO" in step 308.

Sub-routine **GEN-CASE-EXAMINED-CASES** starts in step 310 in Fig. 7H. This sub-routine finds generalisation patterns between the input-case from <u>Case-file</u> and cases in the Examined-case-list, which it receives as inputs in step 311. Step 312 determines whether the Examined-case-list is empty and returns in step 313 if this is affirmative. IF the Examined-case-list has items, the sub-routine continues to step 314 where the first case from the Examined-case-list is retrieved. Step 315 calls subroutine **GET-GEN-PATTERN** to calculate a generalisation pattern, Gen-pattern, between the input-case and the case from the Examined-case-list. Subroutine **GET-GEN-PATTERN** starts at step 330 in Fig. 7I.

Step 316 determines whether a Gen-pattern has been found. If a Gen-pattern has been found, the sub-routine determines in step 317 whether the Gen-pattern match any item in the GPList by calling sub-routine **IF-MATCH**. Sub-routine **IF-MATCH** starts at step 350 in Fig. 7J. If the Gen-pattern does not match any item in the GPList, then step 318 adds Gen-pattern to GPList as a new item and continues to step 319. If step 317 find that the Gen-pattern match an item in the GPList, then the sub-routine continues to step 319 where it is determined whether all cases in the Examined-case-list have been considered.

If step 319 determines that cases in the Examined-case-list remain to been considered, then step 320 retrieves the next case from the Examined-case-list and continues to step 315. Alternatively, step 321 outputs the GPList and the sub-routine **GEN-CASE-EXAMINED-CASES** returns in step 322.

Referring to Fig. 7I, wherein sub-routine **GET-GEN-PATTERN** for identifying a generalization pattern, Gen-pattern, between an input-case from the <u>Case-file</u> and one case from the Examined-case-list, is shown. The sub-routine starts in step 330 by obtaining input-case and a case from the Examined-case-list as inputs in step 331.It takes these two cases as input and compares their features. If some features are shared by the two cases, then these shared features will be included in the intersection part of Gen-pattern. The occurrence in Gen-pattern will be 2. Otherwise, if no features are shared between the cases, an empty Gen-pattern will be produced as output.

Step 332 retrieves the first feature from the input-case, named First-feature, followed by step 333 where the first feature from the case from the Examined-case-list, named Second-feature, is retrieved. Step 334 calls sub-routine **FEATURE-SAME** to determine whether First-feature is the same as Second-feature. Sub-routine **FEATURE-SAME** starts at step300 in Fig. 7G. If they are the same, then step 335 would keep this feature in the intersection part of Gen-pattern and proceeds to step 338. If step 334 determined that the features were not the same, then step 336 determines whether all the features of the case from the Examined-case-list have been considered. If this is affirmative, then the sub-routine continues to step 338. If not, then step 337 retrieves the next feature of the case from the Examined-case-list, names it Second-feature and continues to step 334. Step 338 determines whether all features of the input-case have been considered. If this is affirmative, then the sub-routine continues to step 340. However, if step 338 determined that all the features of the input-case have not been considered, then step 339 retrieves the next feature of the input-case, names it First-feature and continues to step 333.

Step 340 determines whether the Gen-pattern is empty. If the Gen-pattern is not empty, then the occurrence of Gen-pattern is made 2 and the subroutine **GET-GEN-PATTERN** returns in step 343 after producing as output Gen-pattern in step 342. If step 340 determines that Gen-pattern is empty, the sub-routine **GET-GEN-PATTERN** also returns in step 343.

Referring to Fig. 7J, sub-routine **IF-MATCH** is shown. This sub-routine is to check whether the intersection of the generalisation pattern of Gen-pattern matches the intersection of GPList. The sub-routine starts in step 350 and accepts Gen-pattern and GPList as inputs in step 351.

Step 352 extracts the intersection part of Gen-pattern and name it Gen-intersection. Step 353 retrieves the first item from GPList and the intersection part of this item is extracted and named GP-intersection in step 354.

Step 355 calls sub-routine **IF-SAME**, which starts at step 370 in Fig. 7K, to determine whether Gen-intersection and GP-intersection are the same. If a match is found, then the sub-routine **IF-MATCH** returns a “YES” in step 359. If a match was not found in step 355, then step 356 determines whether all the items of GPList have been considered. If this is affirmative, then the sub-routine **IF-MATCH** returns a "NO" in step 360. However, If step 356 determines that there are items of GPList that were not yet considered, then step 357 retrieves the next item from the GPList, step 358 extracts the intersection part of the item and name it GP-intersection and the sub-routine continues to step 355.

Referring to Fig. 7K, subroutine **IF-SAME** is shown which check whether Gen-intersection and GP-intersection, obtained as inputs in step 371, are the same. Step 372 retrieves the first feature from Gen-intersection, named First-feature, followed by step 373 where the first feature from GP-intersection, named Second-feature, is retrieved. Step 374 calls sub-routine **FEATURE-SAME** to determine whether First-feature is the same as Second-feature. Sub-routine **FEATURE-SAME** starts at step300 in Fig. 7G. If they are the same, then step 378 determines whether all the features of Gen-intersection have been considered. If this is affirmative, then the sub-routine continues to step 379. If not, then step 388 retrieves the next feature from Gen-intersection, names it First-feature and continues to step 373. However, if step 374 determined that First-feature is not the same as Second-feature, then step 375 determines whether all the features of the GP-intersection have been considered. If this is affirmative, then the sub-routine **IF-SAME** returns a "NO" in step 292. If features of the GP-intersection remain, then step 376 retrieves the next feature from GP-intersection, names it Second-feature and continues to step 374.

Step 379 retrieves the first feature from GP-intersection, named First-feature, followed by step 380 where the first feature from Gen-intersection, named Second-feature, is retrieved. Step 381 calls sub-routine **FEATURE-SAME** to determine whether First-feature is the same as Second-feature. If they are the same, then step 385 determines whether all the features of GP-intersection have been considered. If this is affirmative, then the sub-routine **IF-SAME** returns a "YES" in step 386. If not, then step 387 retrieves the next feature from GP-intersection, names it First-feature and continues to step 380. However, if step 381 determined that First-feature is not the same as Second-feature, then step 382 determines whether all the features of the Gen-intersection have been considered. If this is affirmative, then the sub-routine **IF-SAME** returns a "NO" in step 384. If features of the Gen-intersection remain, then step 383 retrieves the next feature from Gen-intersection, names it Second-feature and continues to step 381.

In Fig. 6, an example of a GPList 500 for a particular viewer is shown, the GPList 500 having been determined by method shown in Figs. 7A to 7K, performed by the learning module 39 of the avatar agent 37 associated with the particular viewer, from a number of previous cases (Cj):

([*Category* = 'drama'], 6)

([*Category* = 'drama', *Sub-category* = 'social', *Year_of_make* = '1999'], 3)

([*Category* = 'drama', *Sub-category* = 'social', *Start_Time* = '2200', *Year_of_make* = '1999'], 2)

([*Sub-category* = 'social'], 8)

([*Sub-category* = 'drama', *Day* = 'Thu'], 2)

([*Category* = 'movie'], 10)

([*Sub-category* = 'social', *Day* = 'Mon'], 2)

([*Category* = 'drama', *Sub-category* = 'social'], 5)

([*Sub-category* = 'social', *Day* = 'Wed'], 2)

The columns 501 in Fig. 6 represent the attributes of the features of the viewer selections and the entries represent the values 502 of the attributes. Each row 503 in the

GPList 500 represents an item in the GPList 500. For example, the third row 503 has the following values and interpretation:

Attribute	**Attribute-value**	**Interpretation**
Category_ID	16	*Category* = 'drama'
Subcategory_ID	564	*Sub-category* = 'social'
Start_Time	2200	*Start_Time* = '2200'
Day	-1	Don't care
EPG_channel	-1	Don't care
Year_of_Make	1999	*Year_of_make* = '1999'

The occurrence of this intersection is 2. All entries with "-1", are interpreted as "Don't care".

The entries from the GPList 500 shown in Fig. 6 are also represented graphically in Fig. 8 wherein a number of occurrences of each of the combinations A to I of attribute-value pairs are shown. Combination A with attribute-value pairs *mood* = 'neutral' and *Category* = 'movie' occurred 10 times in the viewers selections of programs to watch. Similarly, the combination E with attribute-value pairs *mood* = 'neutral', *Category* = 'movie', *Sub-category* = 'social' and *year-of-make* = '1999' occurred only 3 times.

Referring to Fig. 9, when the recommendation module 40 is instructed by the avatar manager 38 to provide television program recommendations, the procedure **RECOMMEND** is started in step 601. As inputs the procedure **RECOMMEND** receives:

- The GPList from the viewer profile database 23, which includes generalisation pattern entries. Each generalization pattern entry includes the intersecting features and the occurrence count;

- A list of program entries and their features (fi) from which to select viewer recommendations 67. This list was retrieved by the avatar manager 38 based on the time/date combination provided by the viewer for which viewer recommendations are required; and
- a predetermined maximum number of recommended programs;

Initially the list of viewer recommendations is empty. If it is determined in step 602 that all the generalization pattern entries have been examined **or** that there are already the predetermined maximum number of viewer recommendations **or** that the list of program entries is empty, then the procedure continues to step 605 where the list of viewer recommendations are passed to the avatar manager 38. Otherwise, the procedure would continue to step 603, wherein a next generalization pattern entry is selected according to a selection criterion. The selection criterion may be one of the following:

- Choose the next generalization pattern entry with the lowest occurrence. (i.e. start from a more specific match).
- Choose the next generalization pattern entry with the highest occurrence (i.e. start from the most general match)
- Choose the next generalization pattern entry with the greatest number of features in the intersection (i.e. the most specific match)

In step 604, all program entries 60 in the list of program entries whose features (fi) includes those in the generalization pattern entry chosen in step 603, are moved from the list of program entries to the list of viewer recommendations. The procedure goes back to step 602.

The methods of Figs 7A to 7K and Fig. 9 may be practiced using a conventional general-purpose computer system 100, such as that shown in Fig. 10 wherein the processes of Figs 7A to 7K and Fig. 9 may be implemented as software, such as an application program executing within the computer system 100. In particular, the steps may be instructions in the software that are carried out by the computer. The software may be stored in a computer readable medium, including the storage devices described below. The software is loaded into the computer from the computer readable medium, and then executed by the computer. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the computer preferably effects an advantageous apparatus in accordance with the embodiments of the invention.

The computer system 100 comprises a computer module 102, input devices such as a keyboard 110 and mouse 112, output devices including a printer 108 and a display device 104.

The computer module 102 typically includes at least one processor unit 114, a memory unit 118, for example formed from semiconductor random access memory (RAM) and read only memory (ROM), input/output (I/O) interfaces including a video interface 122, and an I/O interface 116 for the keyboard 110 and mouse 112. A storage device 124 is provided and typically includes a hard disk drive 126 and a floppy disk drive 128. A magnetic tape drive (not illustrated) may also be used. A CD-ROM drive 120 is typically provided as a non-volatile source of data. The components 114 to 128 of the computer module 102, typically communicate via an interconnected bus 130 and in a manner which results in a conventional mode of operation of the computer system 100 known to those in the relevant art. Examples of computers on which the embodiments can be practised include IBM-PC's and compatibles, Sun Sparcstations or alike computer systems evolved therefrom.

Typically, the application program of the preferred embodiment is resident on the hard disk drive 126 and read and controlled in its execution by the processor 114.

Intermediate storage of the program may be accomplished using the semiconductor memory 118, possibly in concert with the hard disk drive 126. In some instances, the application program may be supplied to the viewer encoded on a CD-ROM or floppy disk and read via the corresponding drive 120 or 128, or alternatively may be read by the viewer from a network via a modem device (not illustrated). Still further, the software can also be loaded into the computer system 100 from other computer readable medium including magnetic tape, a ROM or integrated circuit, a magneto-optical disk, a radio or infra-red transmission channel between the computer module 102 and another device, a computer readable card such as a PCMCIA card, and the Internet and Intranets including email transmissions and information recorded on websites and the like. The foregoing is merely exemplary of relevant computer readable mediums. Other computer readable mediums may be practiced without departing from the scope and spirit of the invention.

The methods of Figs 7A to 7K and Fig. 9 may alternatively be implemented in dedicated hardware such as one or more integrated circuits performing the functions or sub functions of Figs 7A to 7K and Fig. 9. Such dedicated hardware may include graphic processors, digital signal processors, or one or more microprocessors and associated memories.

The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiment being illustrative and not restrictive.

Claims: **The claims defining the invention are as follows:-**

1. A method of forming a database of user viewing characteristics to be used in enabling a subsequent selection, by a corresponding user, of a program for viewing in a television system in which title information and characteristics of programs to be transmitted in future is transmitted in advance as at least one program guide list, the method comprising the steps of:

recording a plurality of characteristics associated with each program viewed by said user;

forming sets of said characteristics, each said set comprising at least two of said characteristics; and

associating at least each set with an ordered value representative of user's desire to view a particular program;

wherein upon entry of a user request for a program recommendation, performing a search of each of said program guide lists for programs with characteristics that best match said sets and notifying said user of an availability of programs that best match said sets as program recommendations.

2. A recommendation system for enabling a selection by a user of a program for viewing in a television system in which title information and characteristics of programs to be transmitted in future is transmitted in advance as at least one program guide list, the method comprising the steps of:

memory means for recording a plurality of characteristics associated with each program viewed by said user;

processing means for forming sets of said characteristics, each said set comprising at least two of said characteristics and for associating each said set with an ordered value representative of said user's desire to view a particular program;

searching means for performing a search of each of said program guide lists for programs with characteristics that best match said sets; and

on-screen display means for notifying said user of an availability of programs that best match said sets as program recommendations upon entry of a user request for program recommendations.

3. A recommendation system as claimed in claim 2, wherein said program recommendations are based on the programs that best match said sets comprising a highest number of said characteristics.

DATED this THIRTIETH day of SEPTEMBER 1999

Canon Kabushiki Kaisha

Patent Attorney for the Applicant
SPRUSON & FERGUSON

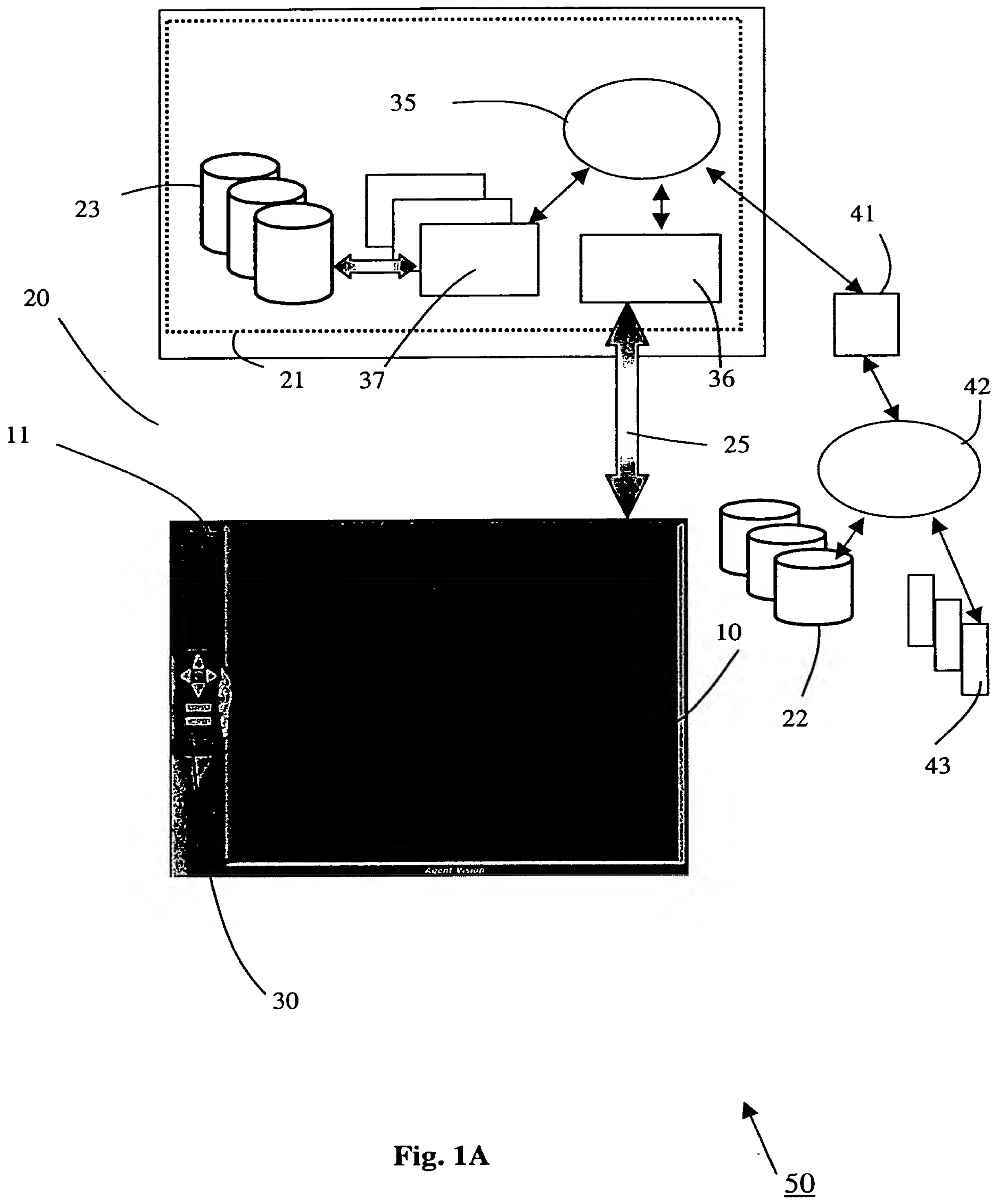


Fig. 1A

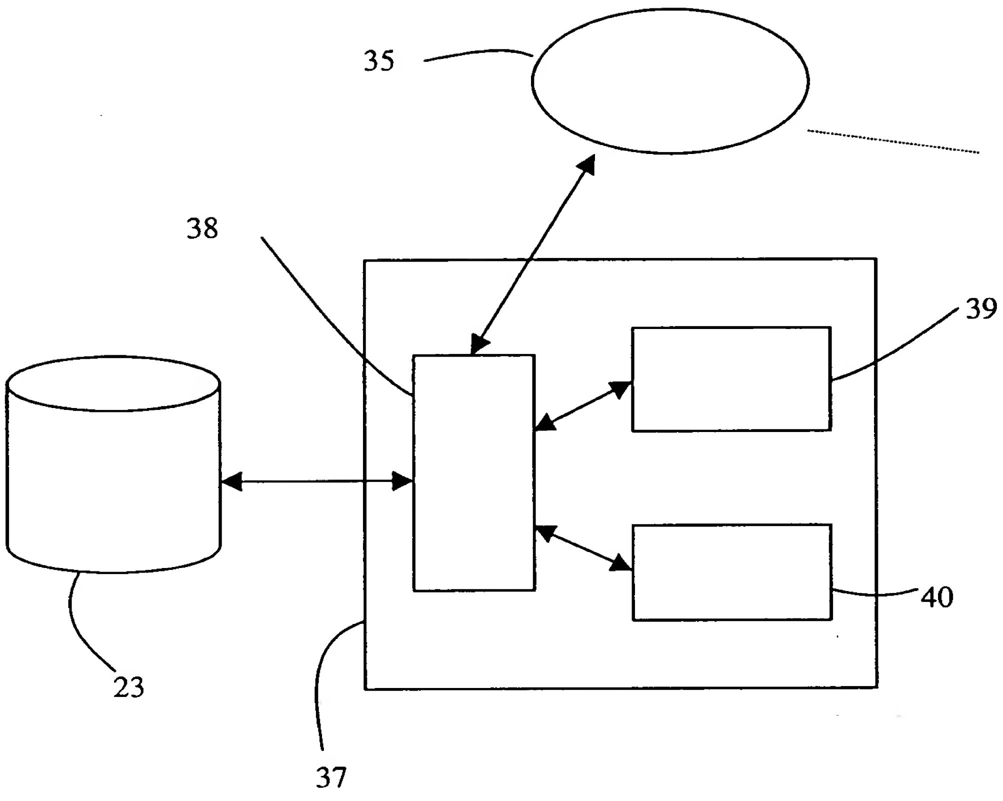


Fig. 1B

61

EPG ID	Program ID	Category ID	Subcategory ID	Title	EPG Channel	Day	Date	StartTime	Exact Duration	Year Of Make
1	1	16	564	ferishitei (felicity) no seishun	Wow	Fri	4/9/99	20:00	50	1999
2	2	16	564	suzuran	Nhk	Mon	4/5/99	8:15	15	1999
3	3	16	217	genroku ryoran	Nhk	Sun	4/4/99	20:00	45	1999
4	4	16	564	nichiyo gekijo/goddo nyu-su	Tbs	Sun	4/11/99	21:00	54	1999
5	5	16	563	abunai ho-kago	Asahi	Mon	4/12/99	20:00	54	1999
6	6	16	564	rippu suteikku (lip stick)	Fuji	Mon	4/12/99	21:00	54	1999
7	7	16	519	romansu	Nihon	Mon	4/12/99	22:00	84	1999
8	8	16	188	furuhata ninzaburo	Fuji	Tue	4/13/99	21:00	69	1999
9	9	16	001	kizudarake no onna	Fuji	Tue	4/13/99	22:15	54	1999
10	10	16	564	happi ai to kando no monogatari	Tokyo	Wed	4/14/99	20:00	54	1999
11	11	16	563	naomi	Fuji	Wed	4/14/99	21:00	54	1999
12	12	16	564	aka ga hashiru - to-gei seishun ki	Nhk	Wed	4/7/99	22:00	45	1999
13	13	16	519	semi daburu	Fuji	Wed	4/14/99	22:00	54	1999
14	14	16	188	Maiko san wa meitantei!	Asahi	Wed	4/15/99	20:00	54	1999
15	15	16	519	koi no kiseki	Asahi	Thu	4/15/99	21:00	69	1999

60

22

Fig. 2

62

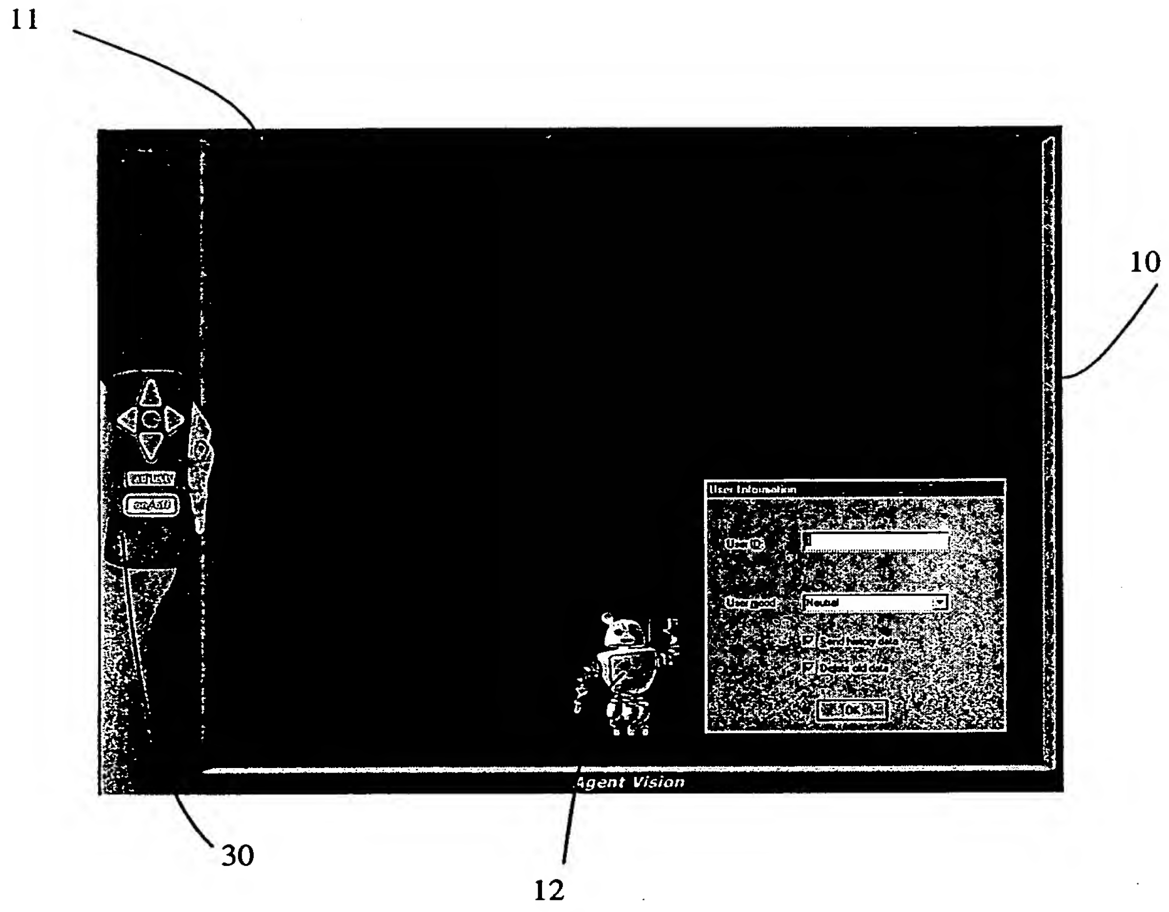

11
10
User Information
User ID
User mood
Neutral
Send history data
Delete old data
OK
Agent Vision
30
12

Fig. 3

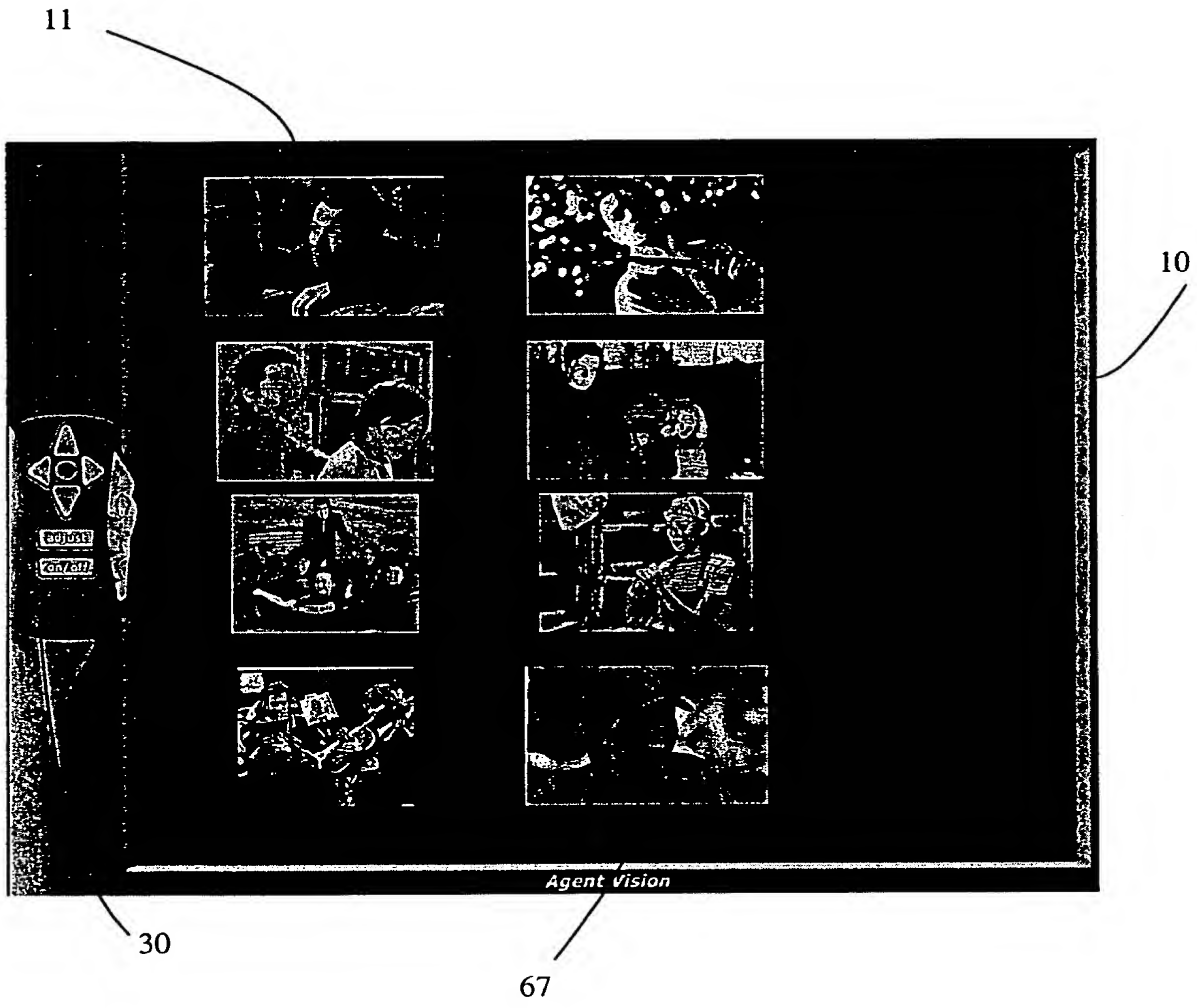


Fig. 4

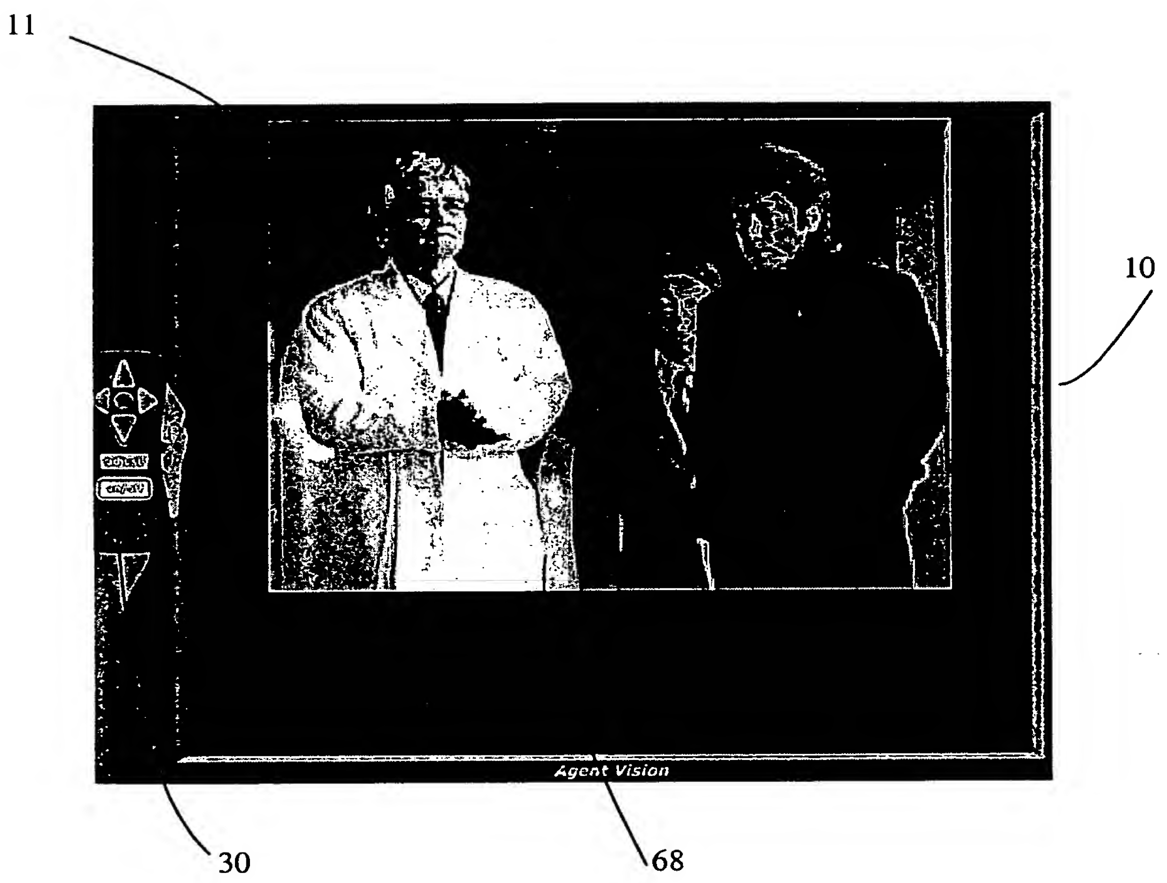
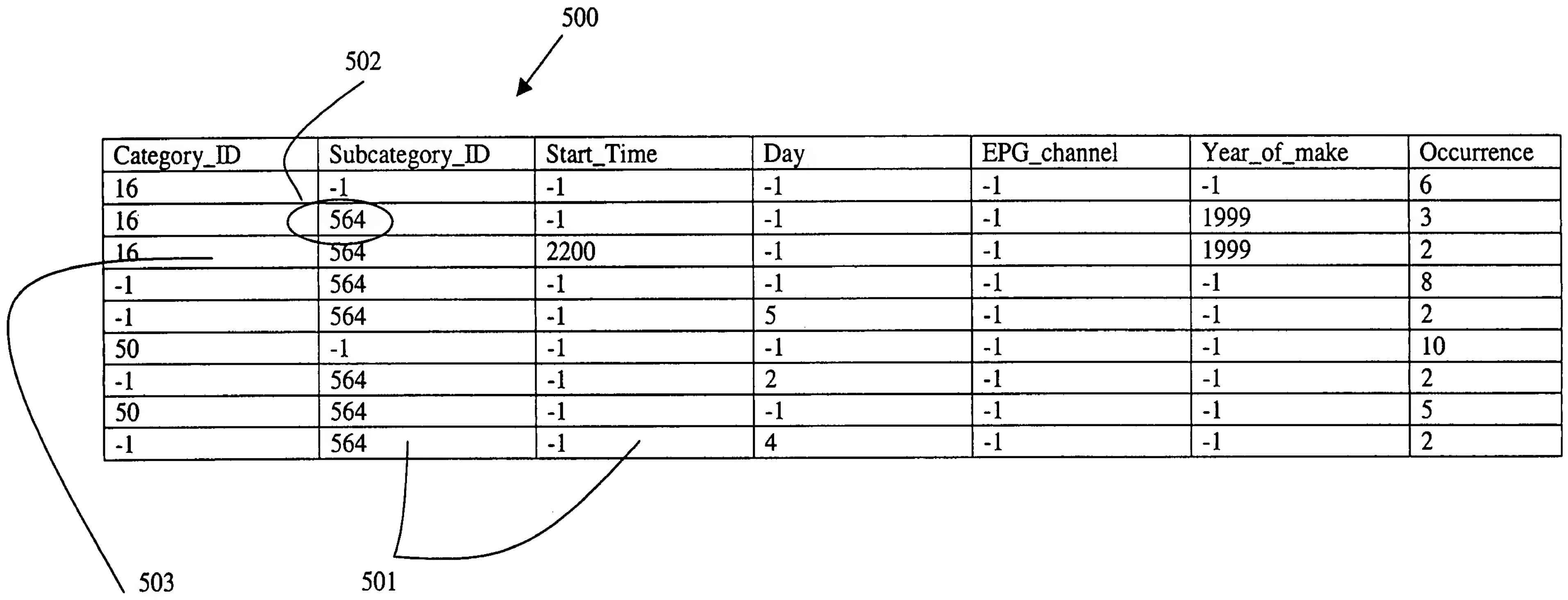


Fig. 5



500
502
503
501

Category_ID	Subcategory_ID	Start_Time	Day	EPG_channel	Year_of_make	Occurrence
16	-1	-1	-1	-1	-1	6
16	564	-1	-1	-1	1999	3
16	564	2200	-1	-1	1999	2
-1	564	-1	-1	-1	-1	8
-1	564	-1	5	-1	-1	2
50	-1	-1	-1	-1	-1	10
-1	564	-1	2	-1	-1	2
50	564	-1	-1	-1	-1	5
-1	564	-1	4	-1	-1	2

Fig. 6

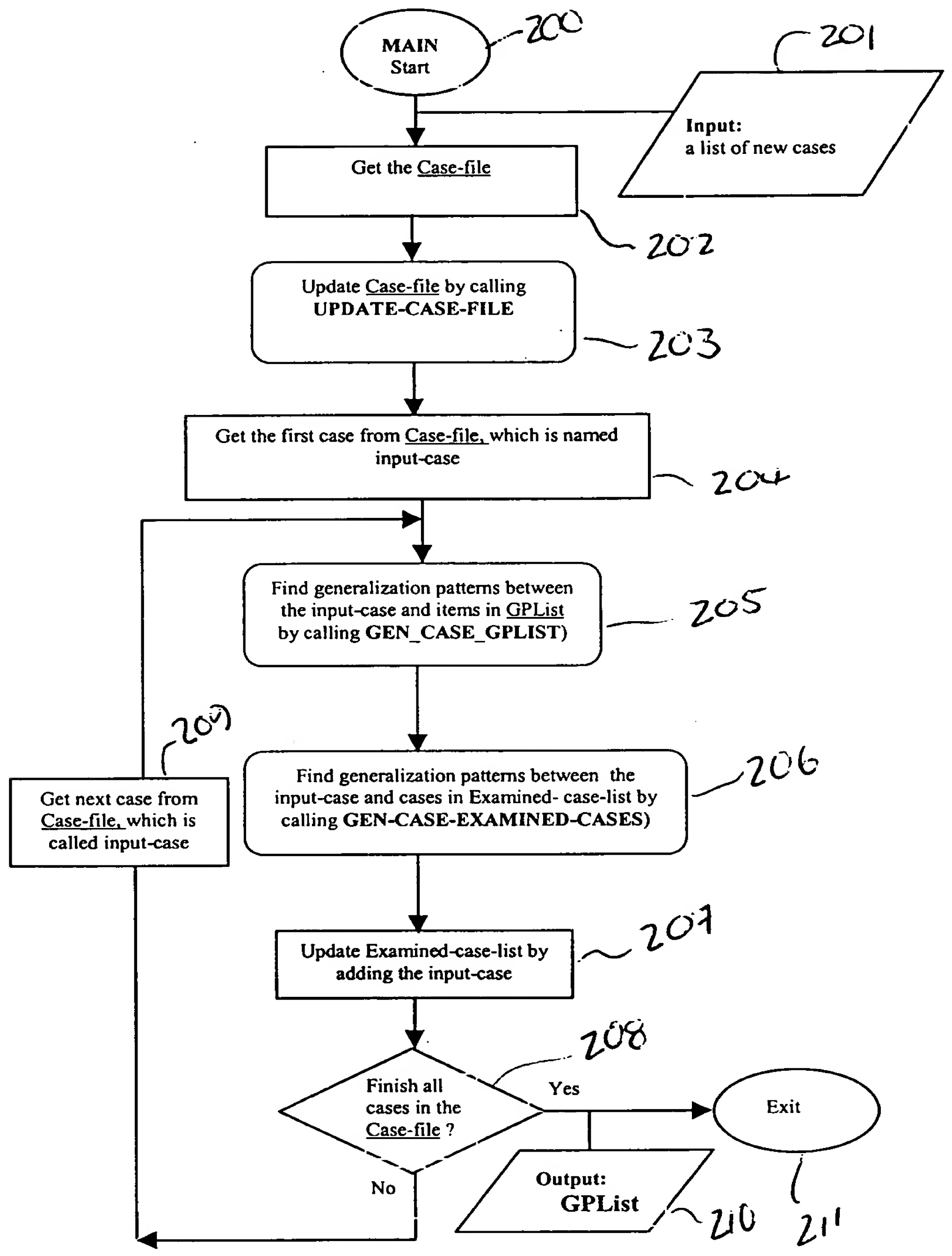
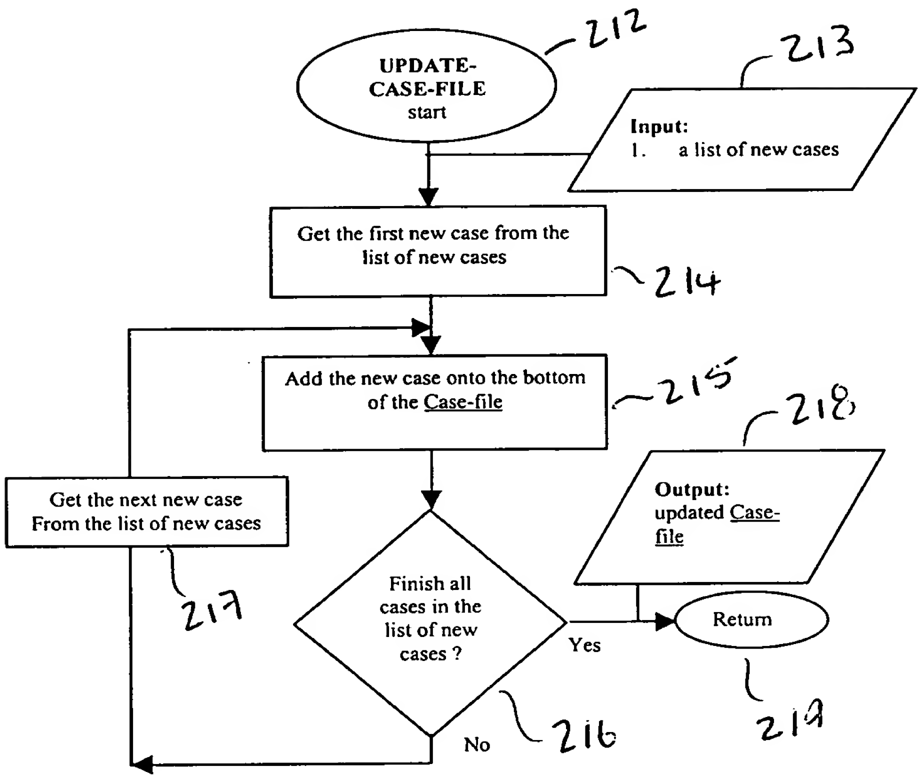
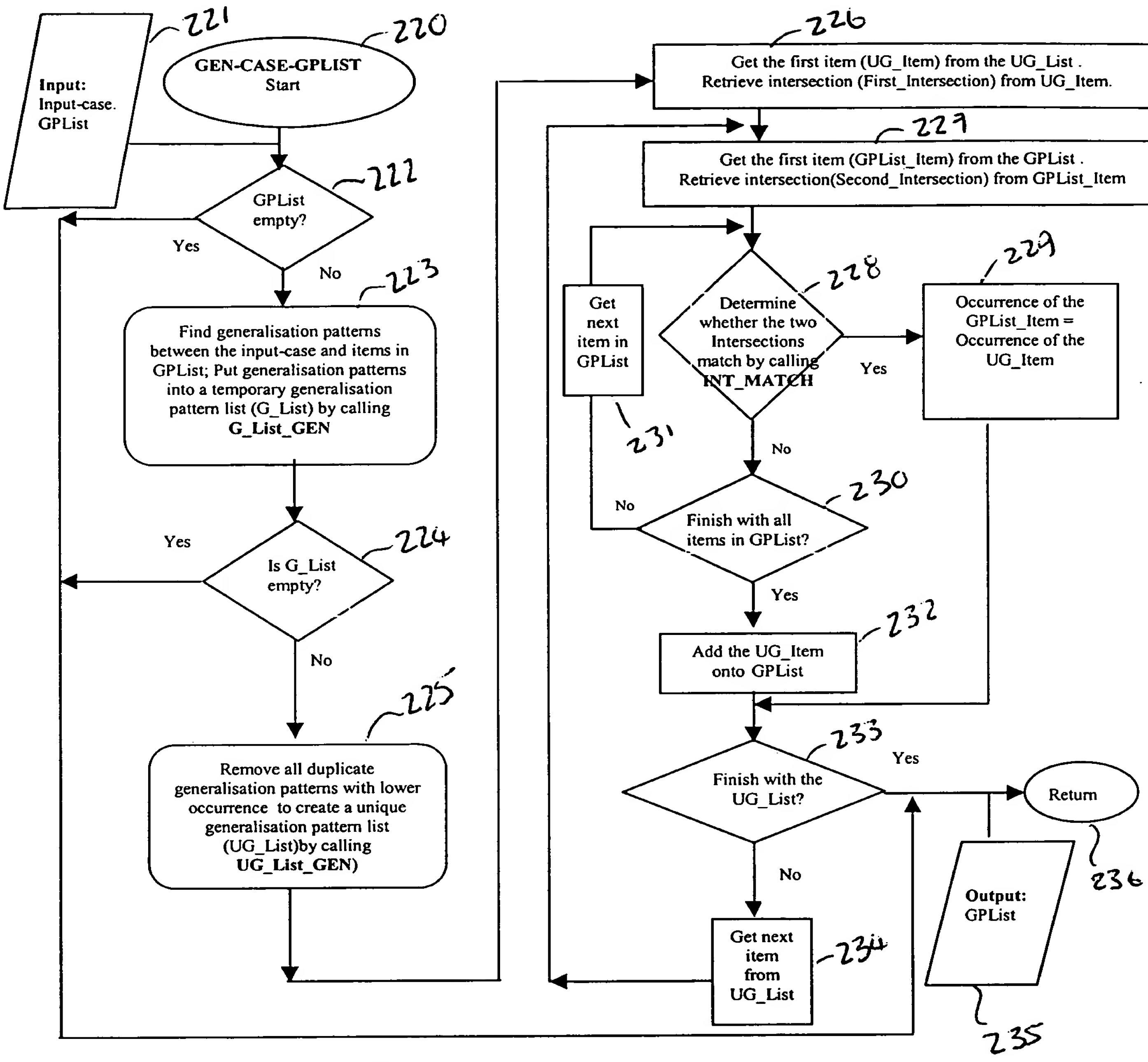

MAIN Start
200
201
Input: a list of new cases
Get the Case-file
202
Update Case-file by calling UPDATE-CASE-FILE
203
Get the first case from Case-file, which is named input-case
204
Find generalization patterns between the input-case and items in GPList by calling GEN_CASE_GPLIST)
205
209
Get next case from Case-file, which is called input-case
Find generalization patterns between the input-case and cases in Examined- case-list by calling GEN-CASE-EXAMINED-CASES)
206
Update Examined-case-list by adding the input-case
207
208
Finish all cases in the Case-file ?
Yes
Exit
No
Output: GPList
210
211

Fig. 7A Procedure **MAIN**



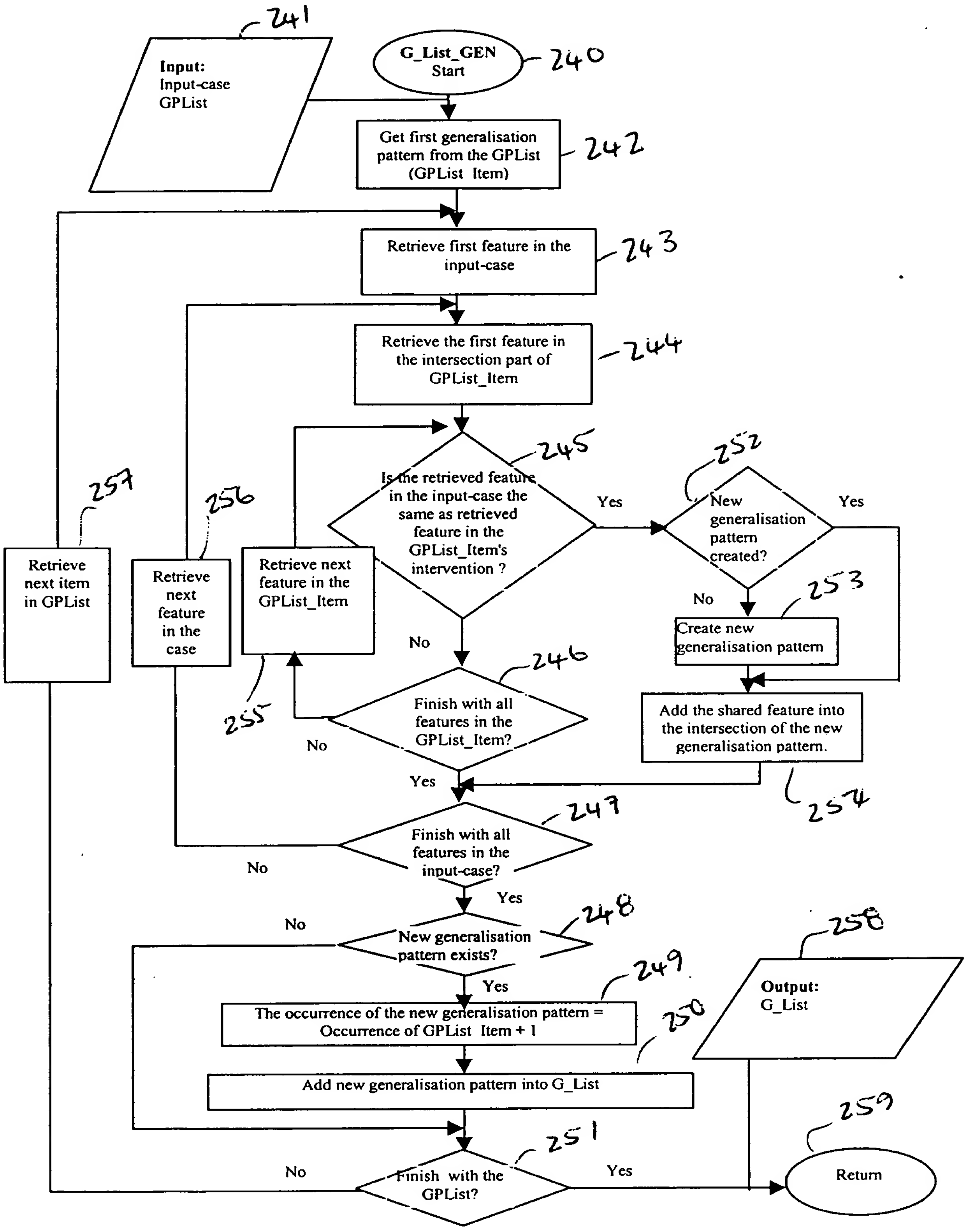
UPDATE-CASE-FILE start
212
Input:
1. a list of new cases
213
Get the first new case from the list of new cases
214
Add the new case onto the bottom of the Case-file
215
218
Output: updated Case-file
Get the next new case From the list of new cases
217
Finish all cases in the list of new cases ?
Yes
Return
No
216
219

Fig. 7B Sub-routine **UPDATE-CASE-FILE**



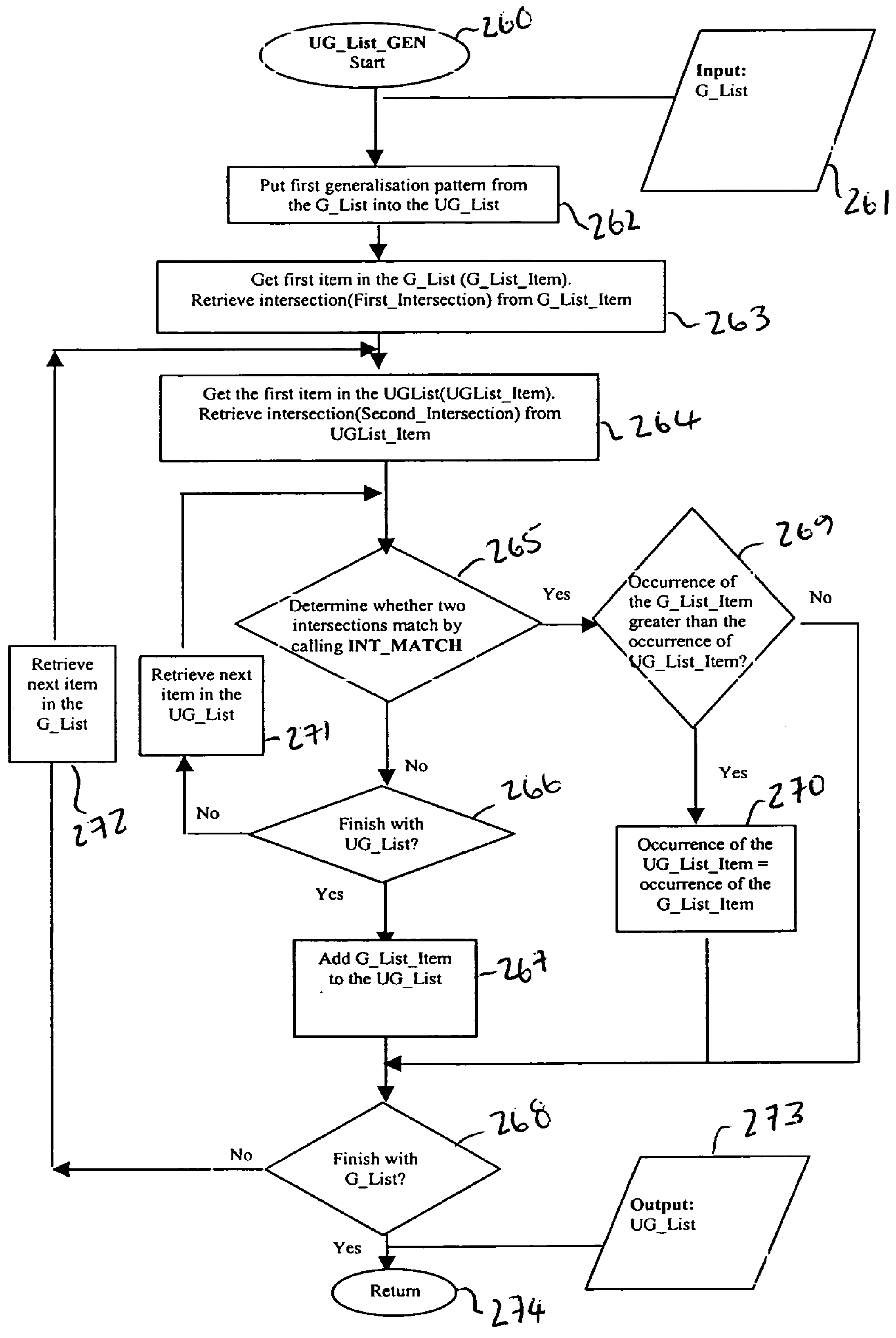
221
220
Input:
Input-case.
GPList
GEN-CASE-GPLIST
Start
GPList empty?
222
Yes
No
223
Find generalisation patterns between the input-case and items in GPList; Put generalisation patterns into a temporary generalisation pattern list (G_List) by calling G_List_GEN
Is G_List empty?
224
Yes
No
225
Remove all duplicate generalisation patterns with lower occurrence to create a unique generalisation pattern list (UG_List)by calling UG_List_GEN)
226
Get the first item (UG_Item) from the UG_List . Retrieve intersection (First_Intersection) from UG_Item.
227
Get the first item (GPList_Item) from the GPList . Retrieve intersection(Second_Intersection) from GPList_Item
Get next item in GPList
231
Determine whether the two Intersections match by calling INT_MATCH
228
Yes
229
Occurrence of the GPList_Item = Occurrence of the UG_Item
No
Finish with all items in GPList?
230
No
Yes
Add the UG_Item onto GPList
232
Finish with the UG_List?
233
Yes
Return
236
No
Output:
GPList
235
Get next item from UG_List
234

Fig. 7C Sub-routine **GEN-CASE-GPLIST**



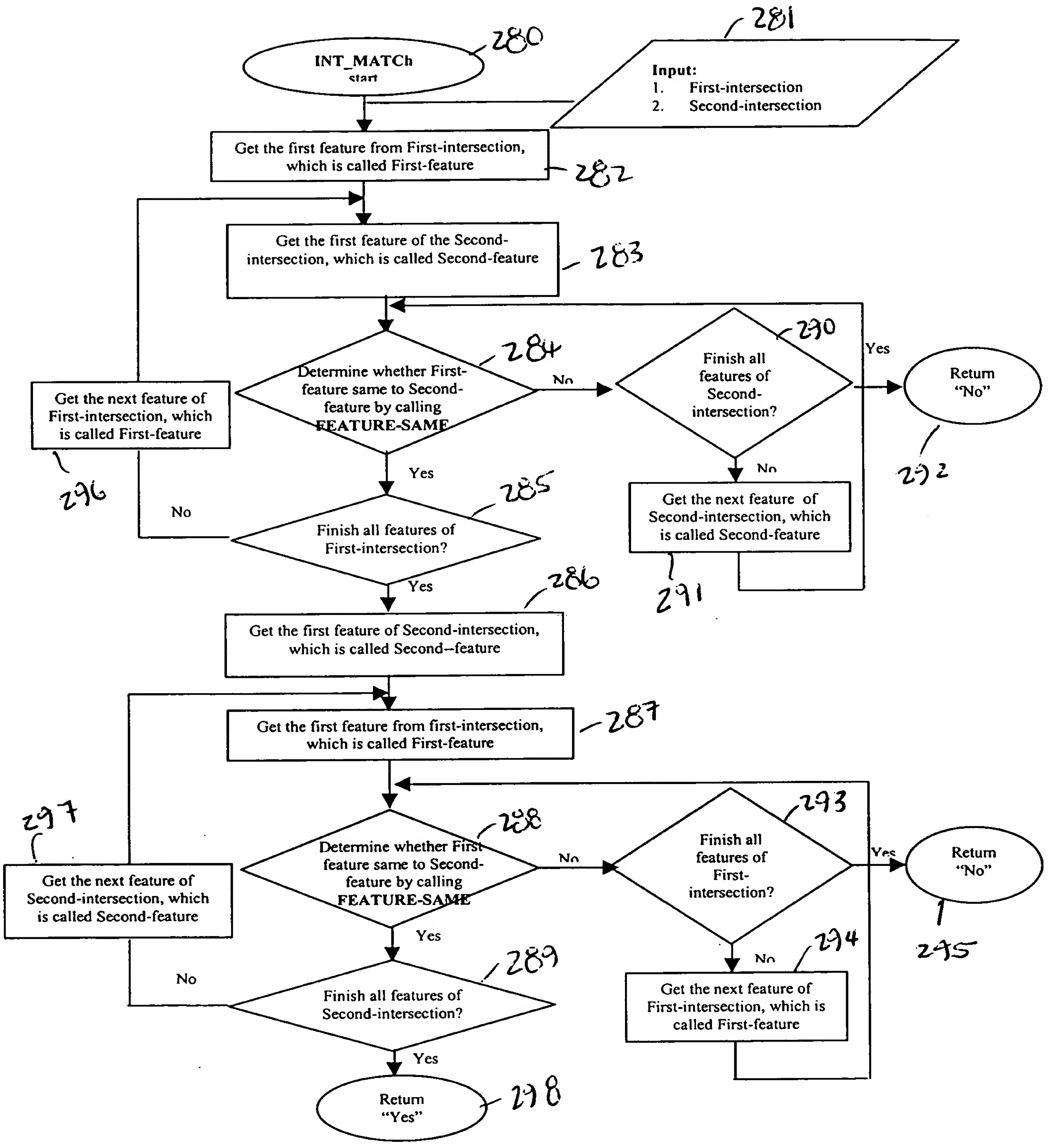
G_List_GEN Start
240
Input: Input-case GPList
241
Get first generalisation pattern from the GPList (GPList Item)
242
Retrieve first feature in the input-case
243
Retrieve the first feature in the intersection part of GPList_Item
244
Is the retrieved feature in the input-case the same as retrieved feature in the GPList_Item's intervention ?
245
Yes
No
New generalisation pattern created?
252
Yes
No
Create new generalisation pattern
253
Add the shared feature into the intersection of the new generalisation pattern.
254
Retrieve next item in GPList
257
Retrieve next feature in the case
256
Retrieve next feature in the GPList_Item
255
Finish with all features in the GPList_Item?
246
No
Yes
Finish with all features in the input-case?
247
No
Yes
New generalisation pattern exists?
248
No
Yes
The occurrence of the new generalisation pattern = Occurrence of GPList Item + 1
249
Add new generalisation pattern into G_List
250
Output: G_List
258
Finish with the GPList?
251
No
Yes
Return
259

Fig. 7D Sub-routine **G_List_GEN**



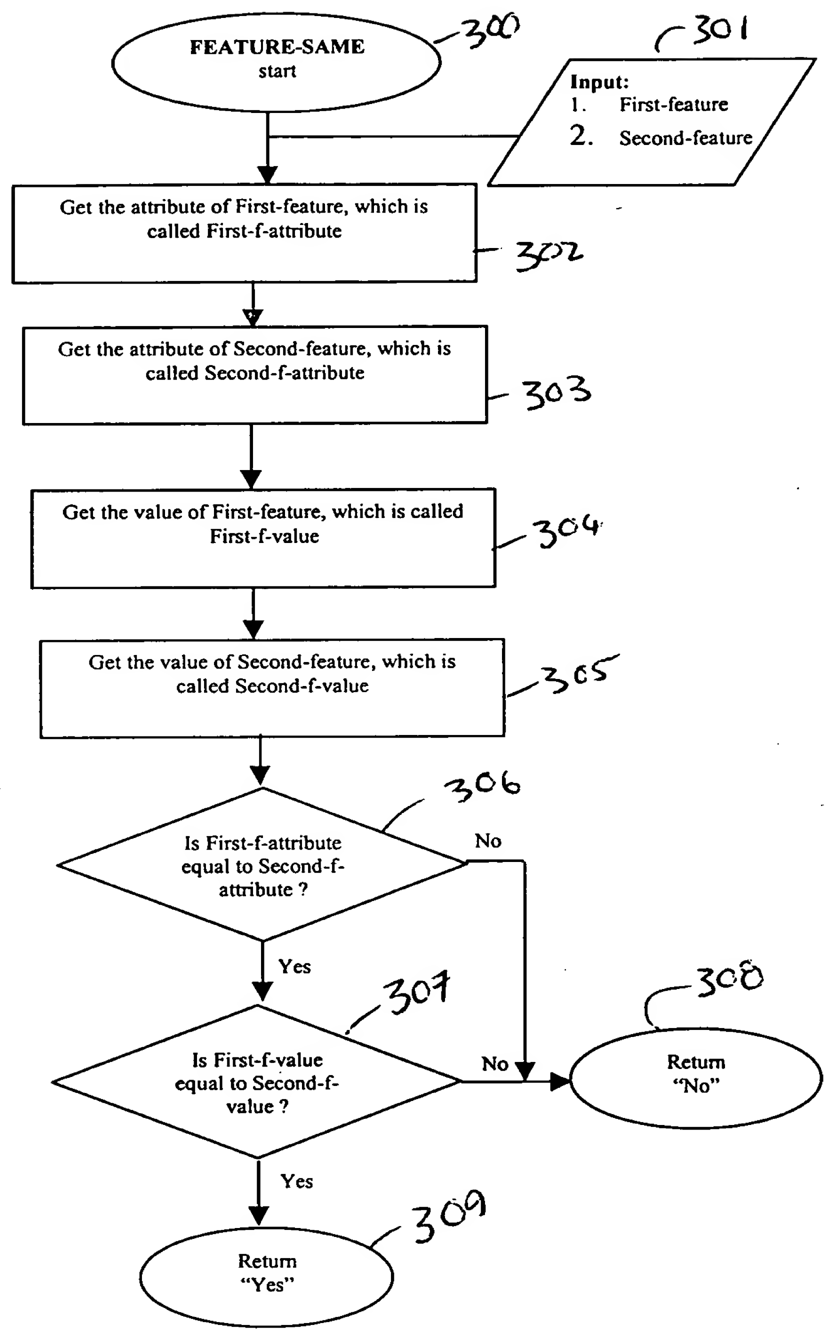
UG_List_GEN Start
260
Input: G_List
261
Put first generalisation pattern from the G_List into the UG_List
262
Get first item in the G_List (G_List_Item). Retrieve intersection(First_Intersection) from G_List_Item
263
Get the first item in the UGList(UGList_Item). Retrieve intersection(Second_Intersection) from UGList_Item
264
Determine whether two intersections match by calling INT_MATCH
265
Yes
Occurrence of the G_List_Item greater than the occurrence of UG_List_Item?
269
No
Yes
Occurrence of the UG_List_Item = occurrence of the G_List_Item
270
No
Finish with UG_List?
266
No
Retrieve next item in the UG_List
271
Retrieve next item in the G_List
272
Yes
Add G_List_Item to the UG_List
267
Finish with G_List?
268
No
Yes
Output: UG_List
273
Return
274

Fig. 7E Subroutine UG_List_GEN



INT_MATCh start
280
Input:
1. First-intersection
2. Second-intersection
281
Get the first feature from First-intersection, which is called First-feature
282
Get the first feature of the Second-intersection, which is called Second-feature
283
Determine whether First-feature same to Second-feature by calling FEATURE-SAME
284
No
Finish all features of Second-intersection?
290
Yes
Return "No"
292
No
Get the next feature of Second-intersection, which is called Second-feature
291
Yes
Finish all features of First-intersection?
285
No
Get the next feature of First-intersection, which is called First-feature
296
Yes
Get the first feature of Second-intersection, which is called Second--feature
286
Get the first feature from first-intersection, which is called First-feature
287
Determine whether First feature same to Second-feature by calling FEATURE-SAME
288
No
Finish all features of First-intersection?
293
Yes
Return "No"
295
No
Get the next feature of First-intersection, which is called First-feature
294
Yes
Finish all features of Second-intersection?
289
No
Get the next feature of Second-intersection, which is called Second-feature
297
Yes
Return "Yes"
298

Fig. 7F Sub-routine INT_MATCH



FEATURE-SAME start
300
301
Input:
1. First-feature
2. Second-feature
Get the attribute of First-feature, which is called First-f-attribute
302
Get the attribute of Second-feature, which is called Second-f-attribute
303
Get the value of First-feature, which is called First-f-value
304
Get the value of Second-feature, which is called Second-f-value
305
306
Is First-f-attribute equal to Second-f-attribute ?
No
Yes
307
Is First-f-value equal to Second-f-value ?
No
Yes
308
Return "No"
309
Return "Yes"

Fig. 7G Sub-routine **FEATURE-SAME**

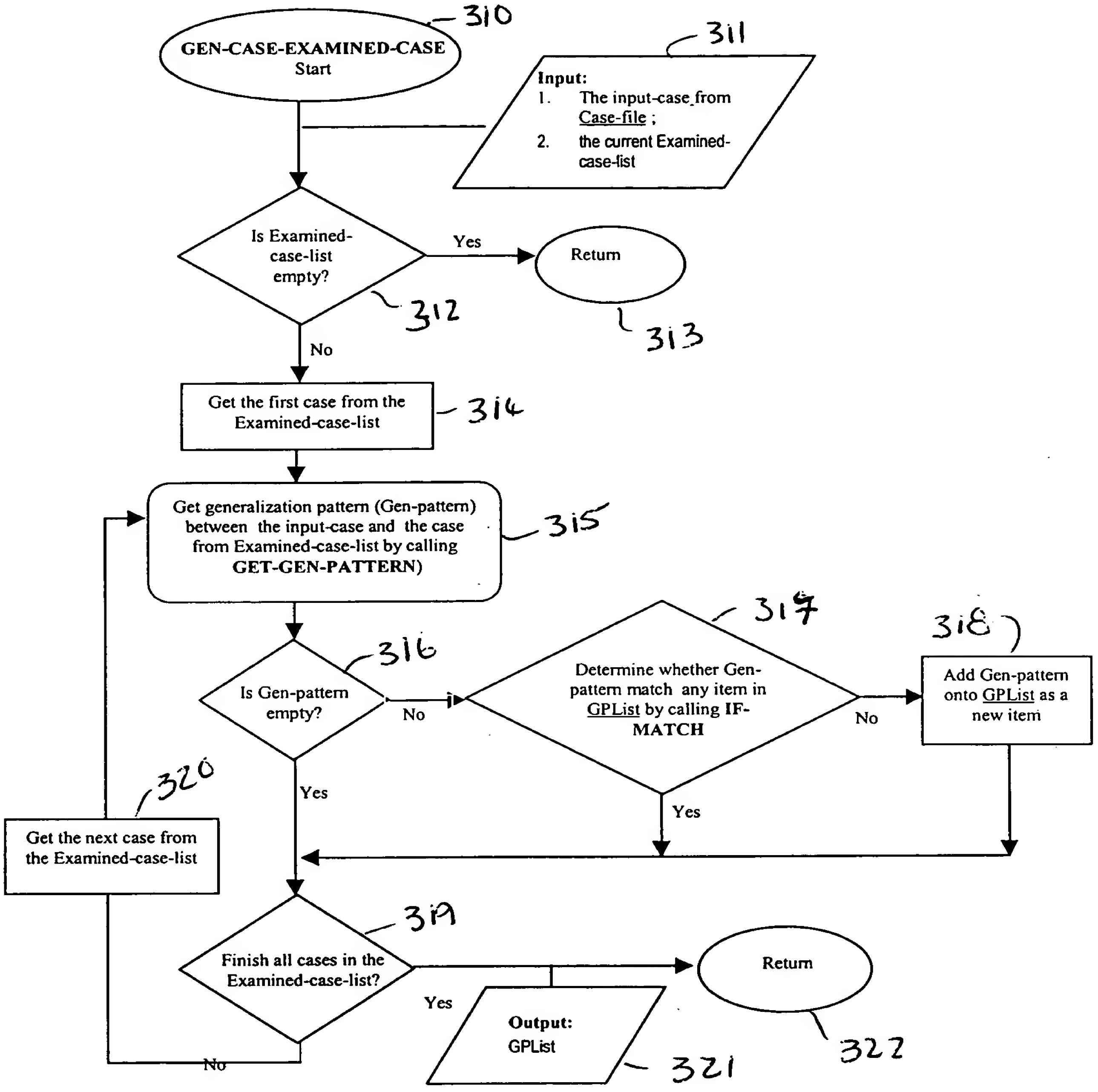

GEN-CASE-EXAMINED-CASE Start
310
Input:
1. The input-case from Case-file ;
2. the current Examined-case-list
311
Is Examined-case-list empty?
312
Yes
Return
313
No
Get the first case from the Examined-case-list
314
Get generalization pattern (Gen-pattern) between the input-case and the case from Examined-case-list by calling GET-GEN-PATTERN)
315
Is Gen-pattern empty?
316
No
Determine whether Gen-pattern match any item in GPList by calling IF-MATCH
317
No
Add Gen-pattern onto GPList as a new item
318
Yes
Yes
Get the next case from the Examined-case-list
320
Finish all cases in the Examined-case-list?
319
Yes
Output: GPList
321
Return
322
No

Fig. 7H Sub-routine **GEN-CASE-EXAMINED-CASES**

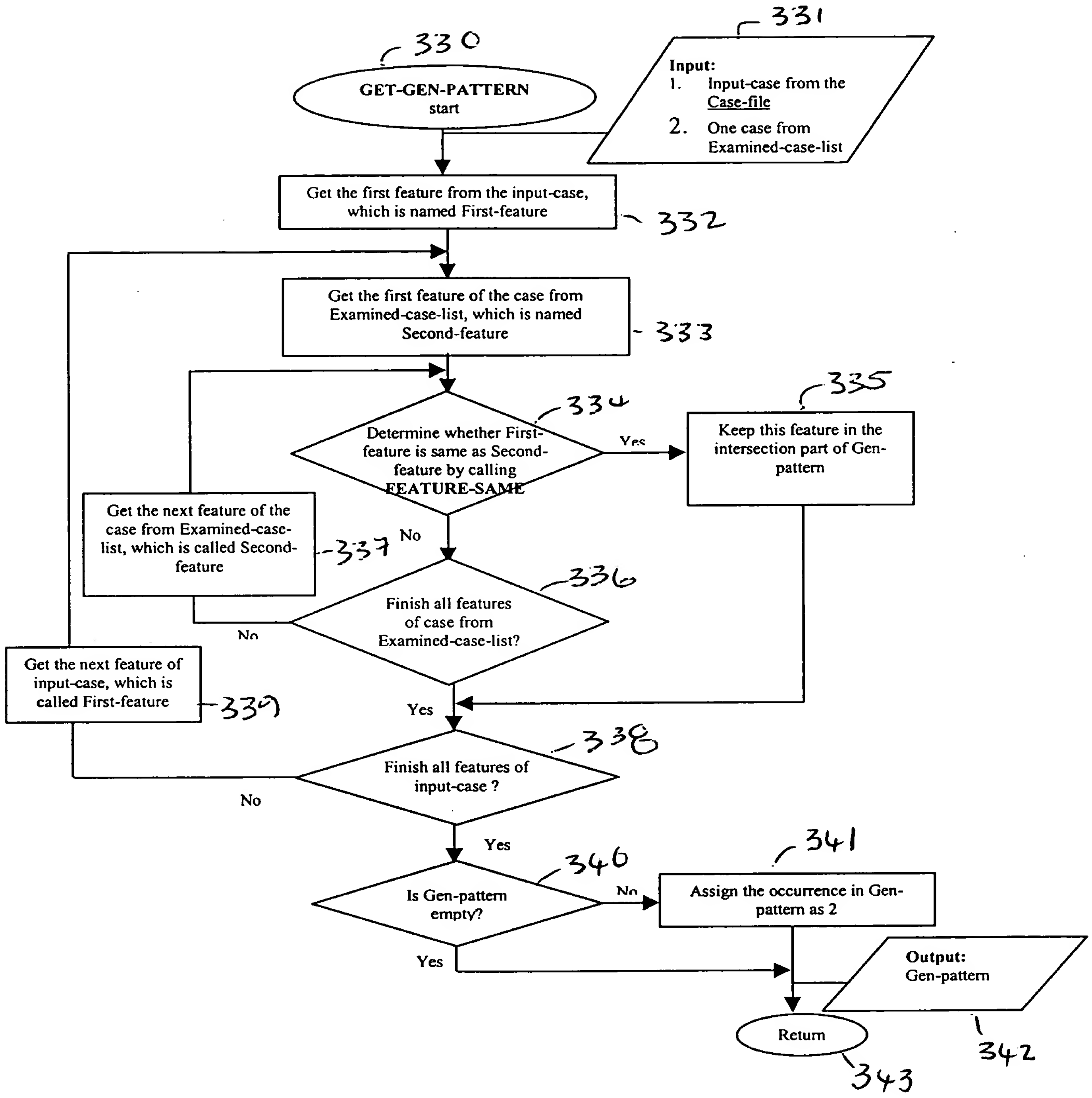

330
GET-GEN-PATTERN start
331
Input:
1. Input-case from the Case-file
2. One case from Examined-case-list
Get the first feature from the input-case, which is named First-feature
332
Get the first feature of the case from Examined-case-list, which is named Second-feature
333
334
Determine whether First-feature is same as Second-feature by calling FEATURE-SAME
Yes
335
Keep this feature in the intersection part of Gen-pattern
No
Get the next feature of the case from Examined-case-list, which is called Second-feature
337
336
Finish all features of case from Examined-case-list?
No
Get the next feature of input-case, which is called First-feature
339
Yes
338
Finish all features of input-case ?
No
Yes
341
340
Is Gen-pattern empty?
No
Assign the occurrence in Gen-pattern as 2
Yes
Output: Gen-pattern
Return
342
343

Fig. 7I Sub-routine **GET-GEN-PATTERN**

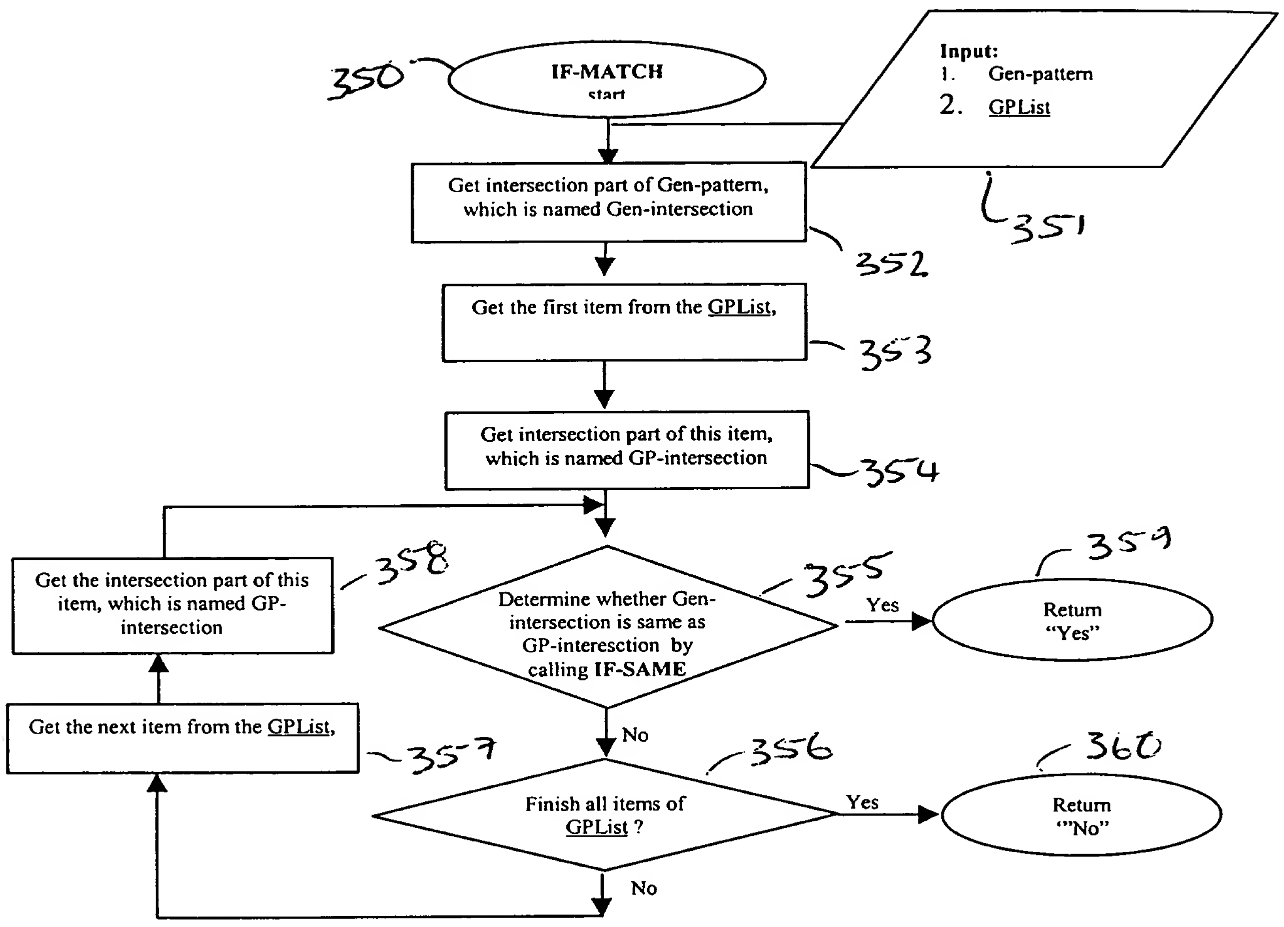

350
IF-MATCH
start
Input:
1. Gen-pattern
2. GPList
351
Get intersection part of Gen-pattern, which is named Gen-intersection
352
Get the first item from the GPList,
353
Get intersection part of this item, which is named GP-intersection
354
Get the intersection part of this item, which is named GP-intersection
358
Determine whether Gen-intersection is same as GP-interection by calling IF-SAME
355
Yes
359
Return "Yes"
Get the next item from the GPList,
357
No
356
Finish all items of GPList ?
Yes
360
Return "No"
No

Fig. 7J Sub-routine IF-MATCH

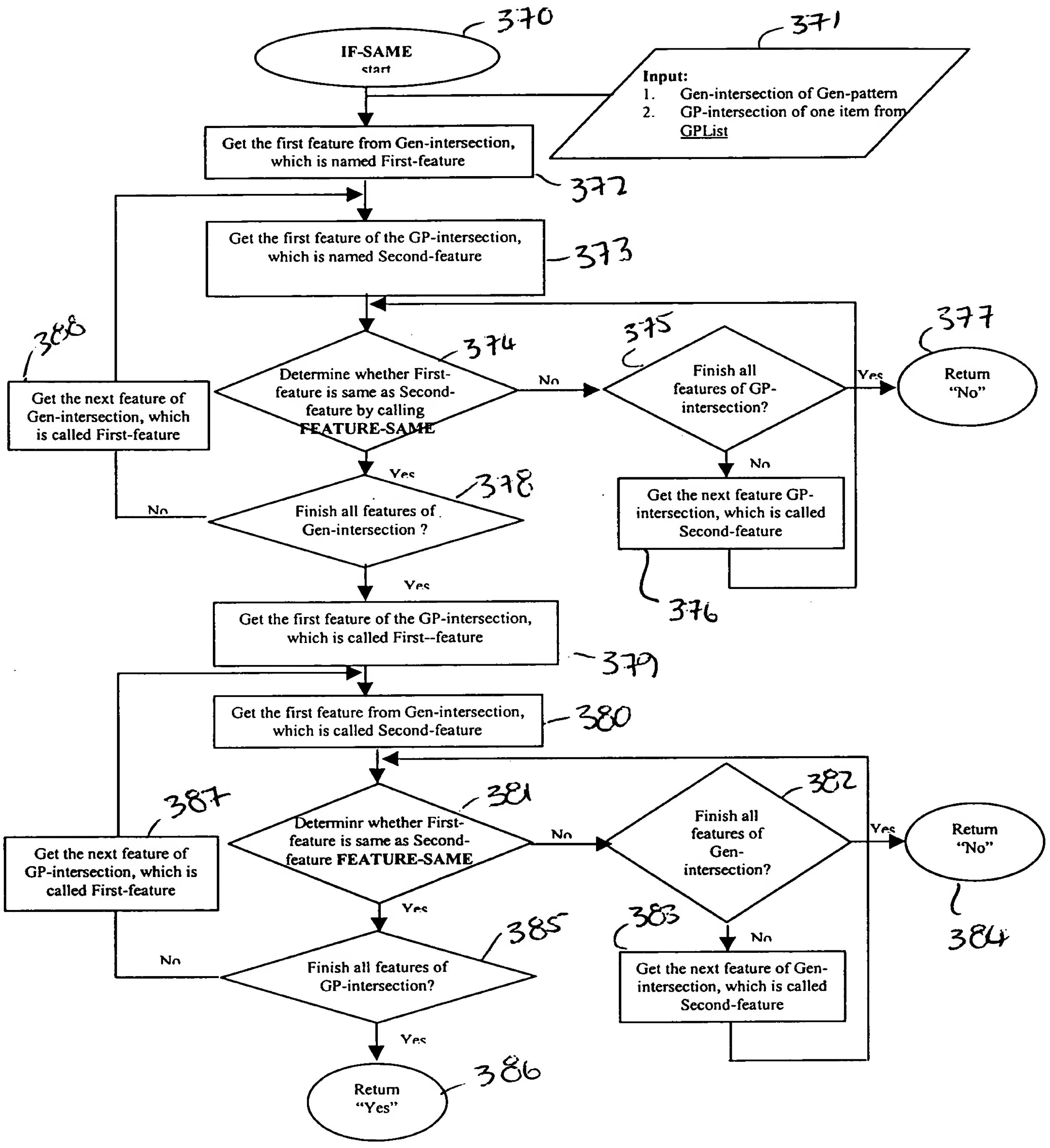
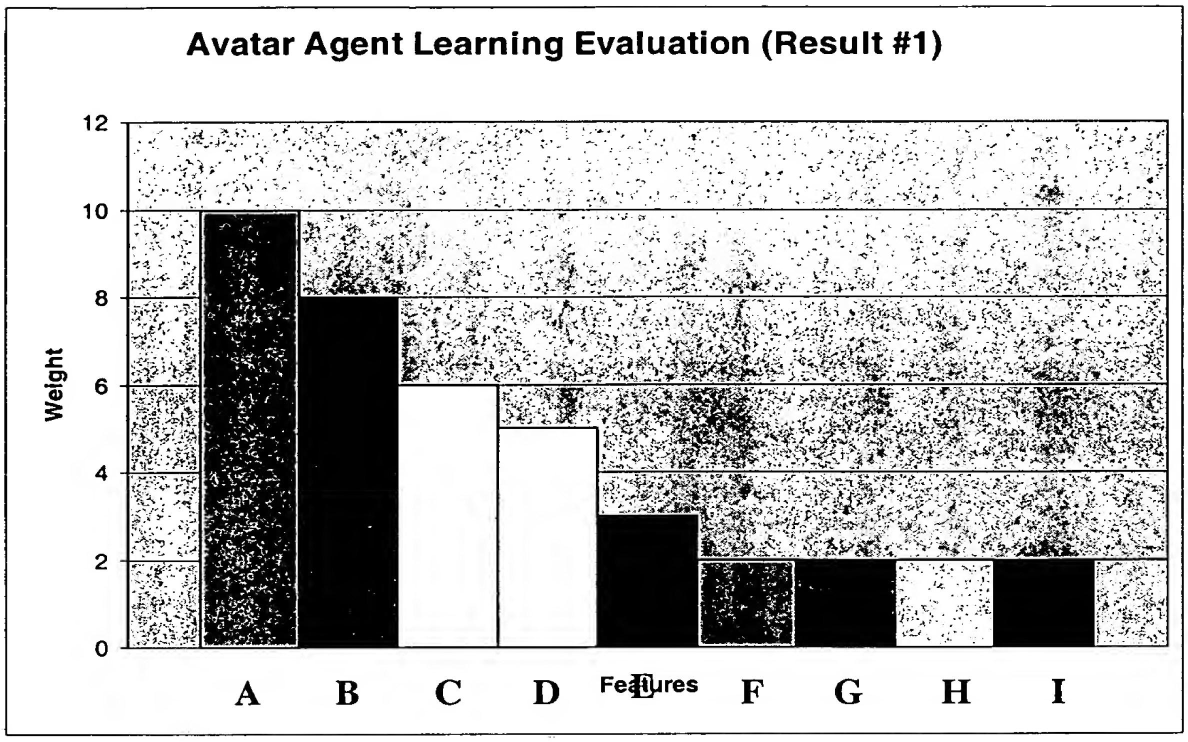

370
IF-SAME start
371
Input:
1. Gen-intersection of Gen-pattern
2. GP-intersection of one item from GPList
Get the first feature from Gen-intersection, which is named First-feature
372
Get the first feature of the GP-intersection, which is named Second-feature
373
388
Get the next feature of Gen-intersection, which is called First-feature
374
Determine whether First-feature is same as Second-feature by calling FEATURE-SAME
No
375
Finish all features of GP-intersection?
Yes
377
Return "No"
No
Get the next feature GP-intersection, which is called Second-feature
376
Yes
378
No
Finish all features of Gen-intersection ?
Yes
Get the first feature of the GP-intersection, which is called First--feature
379
Get the first feature from Gen-intersection, which is called Second-feature
380
387
Get the next feature of GP-intersection, which is called First-feature
381
Determinr whether First-feature is same as Second-feature FEATURE-SAME
No
382
Finish all features of Gen-intersection?
Yes
Return "No"
384
No
383
Get the next feature of Gen-intersection, which is called Second-feature
Yes
385
No
Finish all features of GP-intersection?
Yes
Return "Yes"
386

Fig 7K Sub-routine IF-SAME



The features extracted by learning are described below:

A: category (=movie)
B: sub-category (=social)
C: category (=drama)
D: category (=movie) + sub-category (social)
E: category (drama) + sub-category (social) + year-of-make (=1999)
F: category (drama) + sub-category (social) + Start_time (22:00) + year_of_make (1999)
G: sub-category (drama) + day (Thursday)
H: sub-category (social) + day (Monday)
I: sub-category (social) + day (Wednesday)

Fig. 8

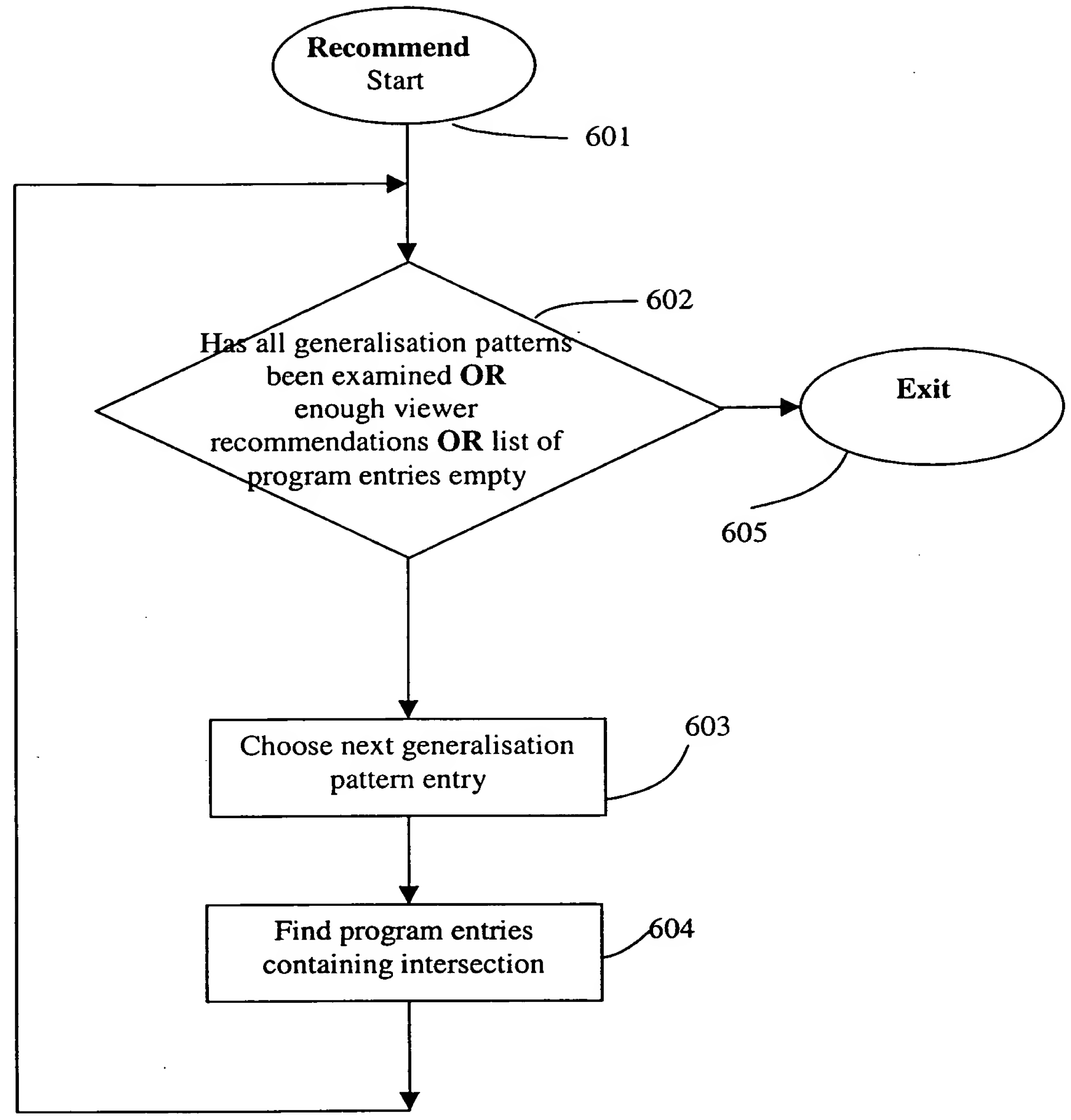

Recommend
Start
601
Has all generalisation patterns been examined **OR** enough viewer recommendations **OR** list of program entries empty
602
Exit
605
Choose next generalisation pattern entry
603
Find program entries containing intersection
604

Fig. 9

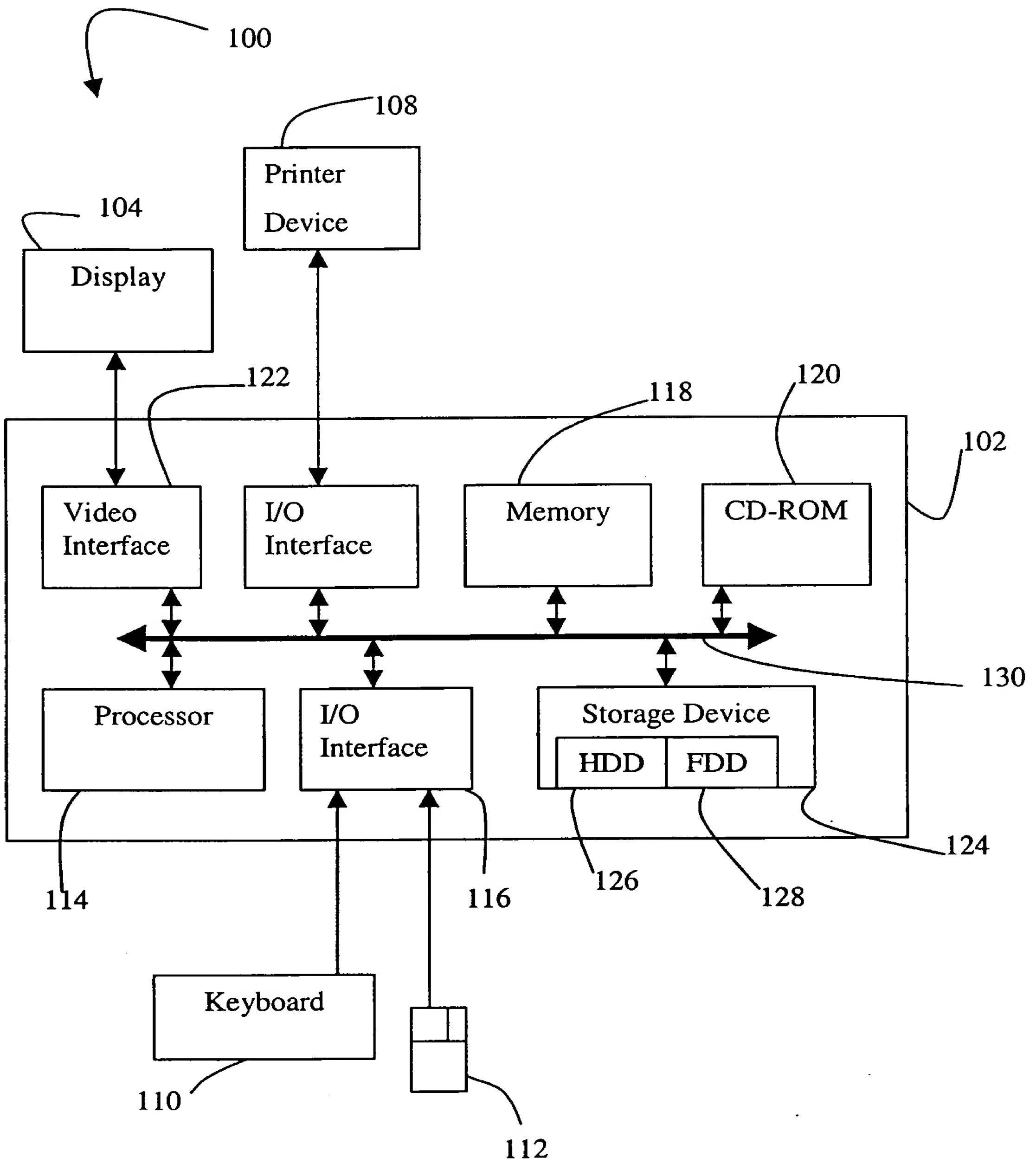

100
108
Printer Device
104
Display
122
118
120
102
Video Interface
I/O Interface
Memory
CD-ROM
130
Processor
I/O Interface
Storage Device
HDD
FDD
124
114
116
126
128
Keyboard
110
112

Fig. 10